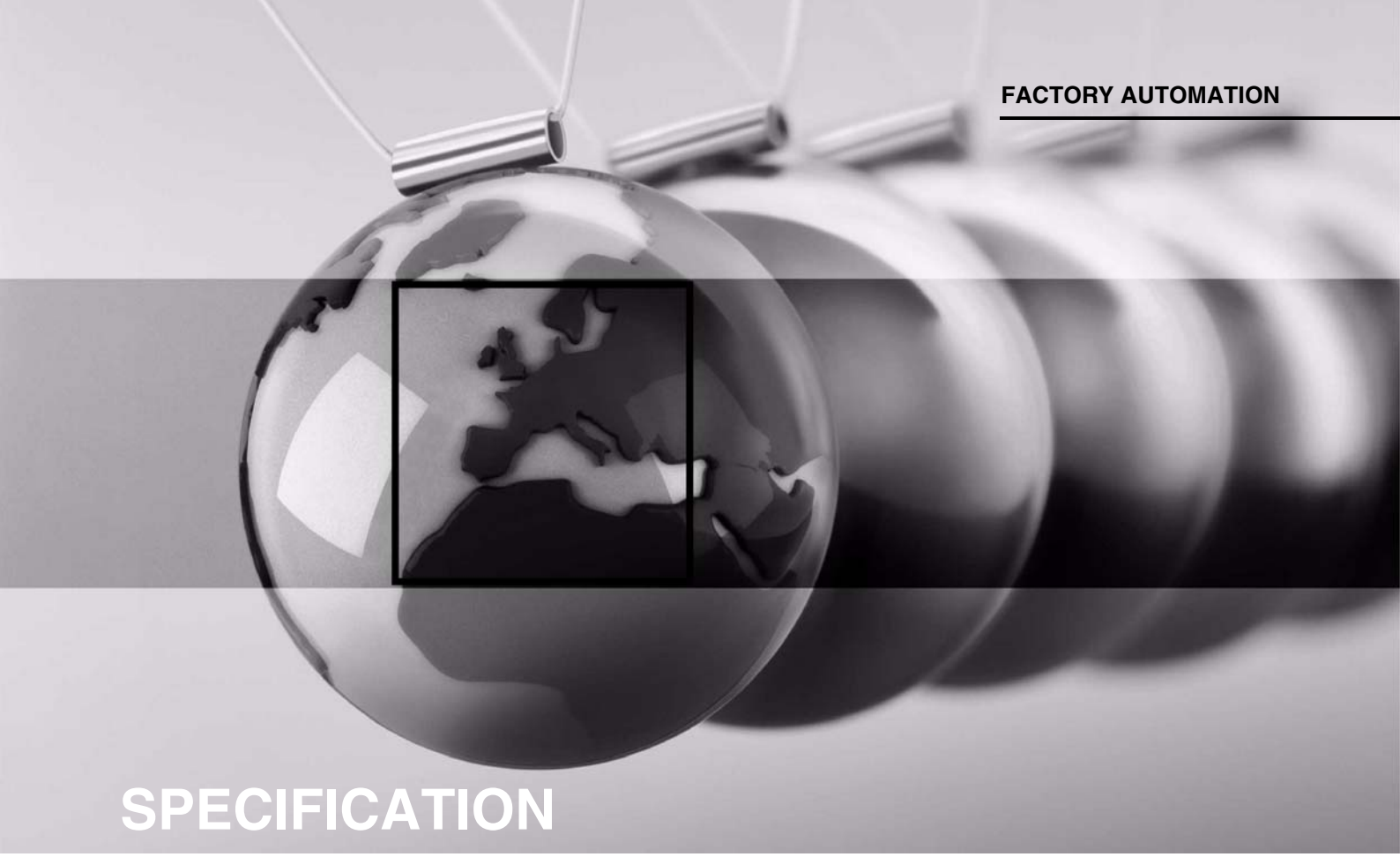

SPECIFICATION

READER-HOST INTERFACE

CLIENT VERSION

PEPPERL+FUCHS
SENSING YOUR NEEDS

With regard to the supply of products, the current issue of the following document is applicable: The General Terms of Delivery for Products and Services of the Electrical Industry, published by the Central Association of the Electrical Industry (Zentralverband Elektrotechnik und Elektroindustrie (ZVEI) e.V.) in its most recent version as well as the supplementary clause: "Expanded reservation of proprietorship"

Table of Contents

1 Scope

This interface control document (ICD) specifies the communication protocol between the Reader and application software that runs on the host computer, specific reader commands, examples of a variety of ways to communicate and send data to the reader (i.e., RS232, USB, RF) and command/communication types.

It is important to note that some functionality has changed and/or been added for each firmware release. Make sure to use the latest released firmware for both boot and app. MAH200/MAH300 default values listed in Section 8 were compiled using firmware version 4634. MAH120 default values listed in Section 8 were compiled using firmware version 4226.

2 Host to Reader Command Overview

This section is intended to introduce developers/users to the basic command types of the reader. There are two ways to send a command to the reader; from a host computer, or by scanning a barcode containing a command sequence. In addition, there are two methods of sending a command from a host computer to the reader; packetized and text commands.

2.1 Packetized Commands

Packetized commands are the most reliable way to communicate to the reader. The packet consists of a prefix and a suffix. The prefix contains the amount of data to be transmitted and the suffix contains error detection. Unlike text commands, packetized commands are always enabled. (See Section 6.2)

2.2 Text Commands

Text commands are provided as an easy way to send a command to a reader but they lack the reliability of packetized commands. In addition, text commands must be enabled. Text commands can easily be sent from a terminal program and uses a %xx (similar to URL encoding) to translate an escape sequence containing a 2-digit hex value to the corresponding single 8-bit ASCII character. This allows non-printable ASCII characters to be entered via the terminal program. Text commands can be sent via the RS232, USB Virtual COM or RF mode by using appropriate communication software. In addition, the developer/user may send text commands by using CRB files. (See Section 0)

2.3 Barcode Commands

The reader will recognize the following sequence within a barcode as a command to the reader:

SOH ‘X’ GS STX *Text-Command* **EOT** (Packet does not contain spaces)

The *Text-Command* portion contains a text command as described above.

Because the Barcode Command is terminated with ASCII EOT, the *Text-Command* may not contain EOT. If the *Text-Command* needs to contain EOT, encode it as %04.

3 Definitions

3.1 Notations

The interface protocol is described as a set of grammars, indicated by different type styles and symbols. These indications are listed in the table below.

Example	Indication	Grammar
Text-Command	Italic type	Syntactic categories (non-terminals)
space	Bold type	Terminal symbols
0xFF	0x prefix indicating hexadecimal	Literal byte values
'X'	Single quotes	Literal ASCII characters
SOH	All caps	Non-printable ASCII characters
PageUp	Key name	key press-release sequence
shift↓	Key plus down arrow	Key-down only
shift↑	Key plus up arrow	Key-up only
esc | tab	Vertical bar	Alternatives (this or that)
$data_{opt}$	$_{opt.}$ (Opt subscript)	Optional terminals and non-terminals
$packet\text{-}type_{nz}$	$_{nz}$ (Nz subscript)	Applies to all packets except *z* type packets
$crc16_{nr}$	$_{nr}$ (Nr subscript)	Applies to packets sent in "non-raw" mode, i.e., in "packet" mode

4 Communication Medium

The Reader communicates with the Host via USB, keyboard wedge (PS/2 or AT), RS232, PC-Card, or Bluetooth Serial Port Profile. The Host includes appropriate hooks and/or drivers to enable two-way communication with the Reader.

5 Reader to Host Communication

The Reader may be configured in raw mode, where no packet framing or check characters are sent, and packet mode. (See section 2.1 and 2.2) The Reader may also be configured to expect an acknowledgement from the host after each packet and automatic retry when no acknowledgement is received. Standard “one-way” mode of operation uses raw packets, no expected response from host, and no automatic retry. Standard “two-way” mode of operation uses packets with framing and checks characters, expects a response from the host, and automatically resends. If no Acknowledgement is received (Ack), three (3) attempts to resend are made.

5.1 Raw Data

Reader to Host communication consists of decoded raw data having no framing or check characters. Raw data is sent with no “end of packet” data (crc16). One-way communication, expects no response from Host and no data is resent.

5.2 Packet Data

Data from the Reader to the Host consist of *packets* as specified below. Packetized data is sent using ACK/NAC protocols with framing and check characters. Packets are delivered asynchronously as graphical codes are read and in response to Host to Reader commands. For keyboard communication (USB keyboard or PS/2 or AT keyboard), all *ascii-character*s are transmitted as *keyboard-sequence*s. For all other communication ports, all *ascii-character*s are transmitted as *ascii-byte*s.

Note: when sending data via the keyboard port, the state of the Caps-Lock is assumed to be “off” (i.e., capital letters are always shifted; lower-case letters are never shifted). The Host shall perform capitalization-translation, if necessary, based on the actual state of the Caps-Lock.

Note: Even though the data size field allows up to 65535 bytes of data in a packet, the actual size of a packet either in raw or in packet mode including data and packet overhead is maximum 16384 bytes.

packet:	*start packet-type*$_{nz}$ *data*$_{opt}$ *end*
start:	*packet-start*$_{nr}$ | *XML-start*
packet-start:	**SOH ‘X’ ‘R’** *protocol-version reader-id packet-number timestamp data-size*
end:	*XML-end*$_{nz}$ *crc16*$_{nr}$
XML-start:	**SOH ‘X’ RS** *tag_response* **‘/’**
XML-end:	**EOT**
tag_response:	**‘ap’**
packet-type:	Single ascii-character in table below
data:	*character*
|	*data character*
protocol-version:	**‘1’**
reader-id:	big-endian 32-bit number
packet-number:	*data-packet-number* | *cmd-packet-number*
data-packet-number:	any byte value in the range [0,7f]; increments with each packet; does not increment with resends; used with *z* and *a* packets only
cmd-packet-number:	any byte value in the range [80-ff]; increments with each packet; does not increment with resends; used with all packets other than *z* and *a*
timestamp (4 bytes):	big-endian 32-bit number, indicates timestamp in seconds (relative to Reader power-up or last time set in Reader) (For all but *z* packets, the timestamp represents the time the packet was sent to the host; for *z* packets, the time the code was read.)

timestamp (3 bytes):	three characters in range [0,127], most significant character first, indicates 21-bit timestamp in seconds (relative to Reader power-up or last time set in Reader) (for all but *z* packets, the timestamp represents the time the packet was sent to the host; for *z* packets, the time the code was read)
data-size:	big-endian 16-bit number indicating size of the *data* field (in bytes)
character:	*byte*
|	*keyboard-sequence*
byte:	any byte value in range [**0x00**,**0xFF**]
keyboard-sequence:	*key*
|	**shift**↓ *key* **shift**↑
|	**alt**↓ *decimal-code* **alt**↑
key:	**~** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **0** | **-** | **=**
|	**q** | **w** | **e** | **r** | **t** | **y** | **u** | **i** | **o** | **p** | **[** | **]** | \
|	**a** | **s** | **d** | **f** | **g** | **h** | **j** | **k** | **l** | **;** | **’**
|	**z** | **x** | **c** | **v** | **b** | **n** | **m** | **,** | **.** | **/**
|	**space**
|	**esc** | **tab** | **shift** | **alt** | **ctrl** | **enter** | **backspace**
|	**f1** | **f2** | **f3** | **f4** | **f5** | **f6** | **f7** | **f8** | **f9** | **f10** | **f11** | **f12**
|	**insert** | **delete** | **home** | **end** | **pageup** | **pagedown**
|	**left** | **right** | **up** | **down** | **keypadenter**
|	*digit*
decimal-code:	*digit* | *digit digit* | *digit digit digit* (range [0,255])
digit:	**keypad0** | **keypad1** | **keypad2** | **keypad3** | **keypad4**
|	**keypad5** | **keypad6** | **keypad7** | **keypad8** | **keypad9**
crc16:	*big-endian 16-bit number* representing crc16 of the packet, calculated over the entire packet, excluding the crc16 itself. (See source files *crc16.[hc]* (Appendix) for details on the crc16 algorithm and polynomials to be used.)
crc14:	two consecutive bytes, each in range [0,127], representing crc16 bitwise AND with 0x7f7f, most significant byte first. The packet crc16 is calculated over the entire packet, excluding the *prefix* and the crc14 itself. (See source files *crc16.[hc]* (Appendix) for details on the crc16 algorithm and polynomials to be used.)

The following *packet-types* are defined:

a	indicates that *data* contains the first part of a decode. A sequence of *a* packets always ends with a *z* packet. The data of all *a* packets in a group and the final *z* packet are concatenated by the host. (Also see the Host-to-Reader ‘R’ command description.)
d	indicates that a command and its associated data were successfully received; *data* optionally contains a null-terminated text message.
e	indicates that a command was not successfully received, i.e., it had a bad type, size, or checksum, and should be re-sent; *data* optionally contains a null-terminated text message.
g	indicates that a group of *z* and/or *p* packets follows, terminated by a *d* or *e* packet (*d* for complete group, *e* for incomplete group)

h — indicates that *data* contains a zero-terminated Bluetooth connection string (of printable ASCII characters): IIII BBBBBBBBBBBB L
where:
IIII is the storage index
BBBBBBBBBBBB is the Bluetooth Device Address (twelve hexadecimal digits)
L is “y” if link key stored, “n” if no link key stored

i — indicates that *data* contains the zero-terminated Reader information string (of printable ASCII characters and TAB) in the following format:
VVVVWWWWXXXXSSSSSSSSSSAOODYYYYHHIIIIJJJJKKKKLLLL<TAB>Z…Z
where:
VVVV is the application firmware version number;
WWWW is the bootloader firmware version number;
XXXX is the radio firmware version number;
SSSSSSSSSS is the Reader’s serial number (ten digits);
A is “A” if running firmware is the application, “B” if bootLoader;
OO is the OEM identifier;
D is “D” if and only if the unit has a keypad and display (otherwise, it may be any other printable ASCII character);
YYYY is the flash file system version number;
HH is the hardware type identifier:
00 simulator
01 MAH100
02 MAH200/MAH300
03 MAH200-based OEM module
04 MAH120
IIII is the hardware version number
JJJJ is the maintenance utility version
KKKK is the operating system kernel version
LLLL is the root file-system version
<TAB> is the ASCII TAB character;
Z…Z is the OEM decoder version: a string of up to 16 printable ASCII characters
At least VVVVWWWWXXXXSSSSSSSSSSA will be present. Depending on hardware type and firmware version, one or more of the other fields (and the tab character) may be omitted. For fields to the left of the tab character, if a given field is present, all fields to the left of it will also be present.

m — indicates that *data* contains a message (comment). The *m* packets are not sent when the Reader is in “raw” mode.

p — indicates that *data* contains a portion of a compressed or uncompressed image

r — indicates that the Reader attempted but failed to read a code. (This packet is sent only if the reader is configured to notify the host on unsuccessful read attempts.)

z — indicates that the packet contains data decoded from a code; *data* contains the data decoded from the code.
The *z* type packets do not use the *XML-start*, *packet-type*, or *XML-end* fields.

In “raw” mode (as opposed to “packet” mode), type **m** packets are not sent, only the decoded data is sent for type **z** packets, and all other packets are sent without the *packet-start* and *crc16* fields. In “packet-mode,” the *packet-start* and *crc16* fields are always sent. (See Figure 1.)

Raw Mode

*'**z**' (data) packet:*

Data

*'**i**' (non-z) packet:*

XML '**i**' response						
XML-start				packet-type	data	XML-end
SOH	**'X'**	**RS**	**'ap/'**	**'i'**	VVVV…	**EOT**

Packet Mode Version 1

*'**z**' (data) packet:*

packet-start								Data	packet-end
SOH	**'X'**	**'R'**	**'1'**	reader ID (4 bytes)	packet number (1 byte)	time stamp (4 bytes)	data size (2 bytes)	data	crc16 (2 bytes)

*'**i**' (non-z) packet:*

packet-start								XML '**i**' response							packet-end
SOH	**'X'**	**'R'**	**'1'**	reader ID (4 bytes)	packet number (1 byte)	time stamp (4 bytes)	data size (2 bytes)	**SOH**	**'X'**	**RS**	**'ap/'**	**'i'**	VVVV…	**EOT**	crc16 (2 bytes)

Packet Mode Version 0

*'**z**' (data) packet:*

packet-start						data	packet-end
SOH	**'X'**	**'R'**	**'0'**	time stamp (3 bytes)	data size (2 bytes)	data	crc14 (2 bytes)

*'**i**' (non-z) packet:*

packet-start						XML '**i**' response							packet-end
SOH	**'X'**	**'R'**	**'0'**	time stamp (3 bytes)	data size (2 bytes)	**SOH**	**'X'**	**RS**	**'ap/'**	**'i'**	VVVVWWWWXXXX SSSSSSSSSP	**EOT**	crc14 (2 bytes)

Figure 1: Example 'z' and 'i' Packets in Raw and Packet Modes

Optionally, whenever the Host receives a packet, the Host will respond by sending a *Y* or *R* packet *(defined in the Host to Reader Communication section)* to the Reader. If the expectResponse option

is enabled in the Reader configuration, the Reader will repeatedly retransmit the packet (a configurable number of times) until it receives a *Y* packet.

If a packet received by the Host has a *packet-type* that is not any of the valid types listed above or has the same *packet-number* as the last processed packet of the corresponding type (command or data), the entire packet – up to and including *end* or until timeout – should be discarded by the Host. If the Host had requested a response, it should reissue the request.

If a packet received by the Host from the Reader fails its CRC, the Host should send an *R* packet to the Reader to request that the packet be resent.

6 Host to Reader Communication

Commands and data from the Host to the Reader are sent in the form of *commands* as specified in this section.

Commands are normally sent in USB Native, RS232, and Bluetooth modes. Commands may not be sent via keyboard modes.

Two command formats are supported: **text-command** and packetized-**command**. Text-command format is applicable to RS232 and Bluetooth modes but only if the Reader is configured to accept this format. Packetized-command format is applicable to all interfaces.

text-command: See Section 6.1.

normal-command: See Section 6.2.

After the host sends each complete command, it should wait for a response packet from the Reader. Expected responses are specified along with the command types in section 6.3. If the Reader responds with an *e* packet or doesn't respond within a reasonable timeout period, the Host should resend the command a reasonable number of times.

6.1 Text Commands

Text commands may be sent to the reader in RS232, USB Virtual COM mode, or RF mode using any serial communications software, e.g., HyperTerminal. Text commands may also be sent via the USB and Serial Downloader programs using files with the .crb file extension. The .crb file contains one command per line in the same format as *text-command.* (See Section 0).

Encoded-data is decoded by the Reader by replacing %xx by a single byte with the value specified by the two hex-digits xx, e.g., **%25** would be replaced by character number 0x25, which is ASCII **%**.

text-command: command-type encoded-data$_{opt}$ carriage-return

command-type: Single ASCII character in the set defined in Section 6.3

encoded-data: encoded-datum

| encoded-data encoded-datum

encoded-datum: printable-character | **%** hex-digit hex-digit

printable-character: any byte value in the range [0x20,0x7e]

hex-digit: **'0'** | **'1'** | **'2'** | **'3'** | **'4'** | **'5'** | **'6'** | **'7'** | **'8'** | **'9'**
| **'A'** | **'B'** | **'C'** | **'D'** | **'E'** | **'F'**
| **'a'** | **'b'** | **'c'** | **'d'** | **'e'** | **'f'**

carriage-return: **0x0d**

In order to eliminate inadvertent commanding of the reader, Text Commands are disabled by default (in firmware version 2216+).

To enable Text Commands requires an initial sequence: *;>PAx* where *x* is as defined in section 8, register setting 41. (Note: 'A' is the ASCII character that corresponds to 41 HEX.)

For example, to send the reader commands by typing commands in HyperTerminal:

```
;>PA1
P(xx)yy
P(xx)yy
~
PA8
```

Where **;>PA1** enables text commands with echo and command responses; **P%xxyy** can be any desired commands; **~** saves the settings just sent (the ~ command saves all but communication-related settings); and **PA8** turns text commands back off (except for the initial sequence). (Note: *'A'* is the ASCII character that corresponds to 41 hex, thus P%418 would be equivalent.)

Note: ;>PA1 is used for interactive text commands. If the commands are to be saved in a file and sent non-interactively, use **;>PA7** instead; this enables text commands but disables echo and command responses. (See Section 6.3, Section 8, and Section 0 for additional information.)

The following two examples can be sent to a Reader in RS232 mode from HyperTerminal by just typing the example text.

Example 1 (make the reader beep/vibrate 3 times):

#%03 *Expected output: should make reader beep/vibrate 3 times*

Example 2 (set reader to continuous-read, near field only):

P(C4)5*Expected output: should set reader to continuous-read, near field only*

6.2 Packetized Commands

Packetized commands consist of packetized data sent from host to reader to configure and cause the Reader to perform certain functionalities (e.g. Code XML rules, and settings). Packetized commands are always enabled, unlike text commands. In addition, they include error detection data, making them more robust than text commands.

normal-command: prefix command-type data-size $data_{opt}$ reserved crc14

prefix: **0xEE 0xEE 0xEE 0xEE**

command-type: Single ASCII character in the set defined in Section 6.3

data: datum

| data datum

datum: any byte value in the range [0,255]

data-size: byte value in range [0,240], which indicates size of data (in bytes)

reserved: **0x00**

crc14: two consecutive bytes, each in range [0,127], representing crc16 bitwise AND with 0x7f7f, most significant byte first. The packet crc16 is calculated over the entire packet, excluding the *prefix* and the crc14 itself. (See source files *crc16.[hc]* (Appendix) for details on the crc16 algorithm and polynomials to be used.)

6.3 Command Types

ESC — Reserved for use by JavaScript applications. This command can be used to notify JavaScript functions from a B-String. No defined function in firmware or Viewer JavaScript.

(The Reader will respond with d or e.)

\# — causes the reader to beep and/or vibrate the specified number of times; *data* contains the number as a single character in the range [0,127].

(The Reader will respond with *d* or *e*.)

Example – beep/vibrate three times: #%03

$ — posts a simulated button event to the reader; *data* contains the event number as a single character. See register 39 in Section 8 for a list of the event numbers.

(The Reader will respond with *d* or *e*.)

Example – read both near and far fields: $%03

(— causes the reader to upload any logged error messages (no *data*)

(The Reader will respond with a *g* packet, zero or more *z* packets, and a final *d* or *e*. Each *z* packet contains a portion of the requested data in its *data* field. Note: this is very similar to the response to the X command; however, *p* packets are not applicable and the *g* and *d*/*e* packets are not suppressed even in raw mode.)

) — causes the Reader to erase its log of error messages (no *data*)

(The Reader will respond with *d* or *e*.)

* — causes the Reader to deactivate all top-level XML rules (no *data*)

(The Reader will respond with *d* or *e*.)

\+ — causes the Reader to activate a XML rule; *data* specifies the rule number as a string of ASCII decimal digits

(The Reader will respond with *d* or *e*.)

, — causes the Reader to send a list of current reader settings (no *data*)

(The Reader will respond with *d* containing a space-separated list of all setting values (in order, expressed as hexadecimal ASCII characters) or with *e*.)

\- — causes the Reader to deactivate a XML rule; *data* specifies the rule number as a string of ASCII decimal digits

(The Reader will respond with *d* or *e*.)

. causes the Reader to flash its LEDs; *data* contains four bytes:

LEDs repetitions onTime offTime

LEDs specifies the state of the LEDs: 1 to turn on, 0 to turn off:

Bit	LED	Applicable Hardware
7	reserved, always 0	MAH200/MAH300
6	left blue	MAH200
5	left green	MAH200
4	left red	MAH200
3	center green	MAH300, MAH120
2	center red	MAH300, MAH120
1	right green	MAH200
0	right red	MAH200

Both onTime and offTime are specified in 1/100th seconds (max 2.55 seconds).

(The Reader will respond with *d* or *e*.)

Example – flash left LED amber 3 times, ½ second on, 1 second off:

.%30%03%32%64

/ toggle a bit (or bits) in a Reader setting; *data* contains a printable ASCII string in the following format: hexadecimal setting number in parentheses followed by a 32-bit signed integer value, expressed in ASCII hexadecimal characters (with optional minus sign) or ASCII decimal characters preceded by the '#' character, e.g., /(2e)1000 or /(2e)#4096; the specified integer is XORed with the existing setting value.

(The Reader will respond with *d* or *e*.)

Note: see Section 8 for possible reader settings.

1 indicates the start of a file download; *data* is empty. This command is followed by a sequence of *2* commands containing the file data and a download-end command (e.g., *5*).

(The Reader will respond with *d* or *e*.)

2 indicates a continuation of a file download; *data* contains the next portion of the file data.

(The Reader will not send any response.)

5 indicates the end of a regular file download; *data* contains the name of the file, which is from 1 to 200 letters, digits, periods, hyphens, and underscores, terminated with ASCII NUL.

(The Reader will respond with *d*, *e*, or *f*.)

Note: This command is supported in firmware version 3100+

9 requests the Reader to delete a file from its storage; *data* contains the file name, terminated with ASCII NUL.

(The Reader will respond with *d* or *e*.)

Note: Supported in firmware version 3100+

: requests the Reader to send configuration *data* to the radio module

(The Reader will respond with zero or more *h* followed by a *d* or an *e*.)

For example, *data* consisting of a single zero byte will cause the radio to disconnect from the host. (See Section 9: Radio Commands.)
This command is valid for Firmware versions 2012+

; no operation (may be used for comments in command files)

< causes the Reader to send a list of saved reader settings (no *data*)

(The Reader will respond with *d* containing a space-separated list of all setting values (in order, expressed as hexadecimal ASCII characters) or with *e*.)

= puts setting directly to Reader's non-volatile memory so that it will take effect upon next reboot; *data* is as defined in the / command; the specified integer replaces the existing setting value.

Note: this command can be used to set communication modes without losing communication during the process.

(The Reader will respond with *d* or *e*.)

This command is valid for Firmware versions 2230+

> causes the Reader to send a string of text to the host as a *z* packet; *data* contains the text to send.

(The Reader will respond with a *z* packet containing the text.)

? programs the JavaScript, gocode, or other license; *data* contains the encrypted license data.

(The Reader will respond with *d* or *e*.)

This command is valid for Firmware versions 2232+

@ causes the reader to reset its internal date/timestamp to the specified time; *data* contains the date and/or time in one of the following formats.

yyyy-mm-dd hh:mm:ss
yyyy-mm-dd hh:mm
hh:mm:ss
hh:mm

Note: the separators are optional; only digits are significant.

(The Reader will respond with *d* or *e*.)

Examples:

Set to midnight:	@00:00
Set to Sept 1, 2005 11:52:02 PM:	@2005-09-01 23:52:02

Note: on units without a battery-backed real-time clock, the date and time will reset to 2000-01-01 00:00:00 upon power-up.)

Command enhanced with version 2526+ to include year/month/day information

A notifies the Reader that the previously sent data were rejected for one of the following reasons:

- the packet was encrypted and the decryption failed;
- the host (XML Modem) is locked to a different reader.

The Reader should indicate to the user that the packet has been rejected; e.g., it may sound error beeps. See related setting 0x12f, notify-of-packet-rejection.

(The Reader will not respond to the host.)

This command is valid for Firmware versions 3280+

B defines the strings the Reader will return (or process internally) in response to stored-command-code events; *data* contains the event number of the stored-command-code as a single byte **(in the range 0x09..0x48)** followed immediately by the associated null-terminated string. B-strings are “performance” strings used to enhance/enable specific reader functions and capabilities.

0x09 – 0x0C: reserved for OEM
0x0D – 0x24: reserved
0x0D Performance String A1
0x0E Performance String A2
0x0F Performance String A3
0x10 Reserved as of firmware version 2526
0x11 Performance String B1
0x12 Performance String B2
0x13 Performance String B3
0x14 Performance String C1
0x15 Performance String C2
0x16 Performance String C3
0x17 Performance String D1
0x18 Performance String D2
0x19 Performance String D3
0x1a reserved
0x1b reserved
0x1c Performance String SB
0x1d Performance String SN
0x1e Performance String SF
0x1f Performance String VB (Obsoleted)
0x20 Performance String VN (Obsoleted)
0x21 Performance String VF (Obsoleted)
0x22 Performance String AB (Obsoleted)
0x23 Performance String AN (Obsoleted)
0x24 Performance String AF (Obsoleted)
0x25 – 0x47: for user applications
0x48 User defined “No Read” value (see register 55 in Section 8)

(The Reader will respond with *d* or *e*.)

Example –B-string setting SXGA Near Field Window:
B%0D%01X%1d%02P%54#1024%04

B-strings 0x0D through 0x24 are restored using J command as of version 2526. Performance strings are included with app download and do not require a separate download as of version 2526.

See “Appendix: B-string Settings” for complete details.

Note: Due to increased image capture speed. VGA support was removed in version 4508 (MAH200/MAH300) and 4180 (MAH120).

Note: MAH120 supports B-strings beginning in version 4189.

G — get setting from Reader; *data* contains a single character (0-255), which is the setting number.

(The Reader will respond with *d* and the setting value as a sequence of 8 ASCII hexadecimal digits or with *e*.)

Example - determine if Rectangular Data Matrix is enabled: G%16

Note: see Section 8 for possible reader settings.

I — requests the Reader to send its information string (no *data*).

(The Reader will respond with *i* or *e*.)

J — requests the Reader to restore settings to factory defaults (no *data*).

(The Reader will respond with *d* or *e*.)

L — requests the Reader to send a list of its stored files; *data* is either empty or contains “1”; hidden files are listed only if “1” is specified. (Hidden files are files whose names begin with a period.)

(The Reader will respond in the same manner as with the ‘(’ command, each *z* packet containing a file name as a NUL-terminated string of printable ASCII characters.)

Note: Supported in firmware version 3100+

N — requests the Reader to delete stored data and images (no *data*).

(The Reader will respond with *d* or *e*.)

O — set a bit (or bits) in a Reader setting; *data* is as defined in the / command; the specified integer is ORed with the existing setting value.

(The Reader will respond with *d* or *e*.)

Note: see Section 8 for possible reader settings.

P — put setting to Reader; *data* is as defined in the / command; the specified integer replaces the existing setting value.

(The Reader will respond with *d* or *e*.)

Example – enable Rectangular Data Matrix (setting 0x16, value 1): P%161

Note: see Section 8 for possible reader settings.

Q — clear a bit (or bits) in a Reader setting; *data* is as defined in the / command.

(The Reader will respond with *d* or *e*.)

Note: see Section 8 for possible reader settings.

R	requests that the previously sent packet be re-sent by the Reader; *data* may specify a maximum packet size the receiver will accept: *data* is either empty or specifies a 16-bit big-endian unsigned integer (2 bytes). If *data* is empty or specifies a size less than 32 (the minimum packet size), the Reader will use its preferred maximum packet size. Otherwise, it will use the specified max packet size (or less) and will fragment data across multiple smaller packets when necessary. (The Reader will respond by resending its previous packet or with *e* if there was no previous packet. If the max data size has changed, it may resend the previous data in a sequence of more than one packet.)
T	requests the current date and time (no *data*) (The Reader will respond with *d* with *data* containing the date and time formatted as yyyy-mm-dd hh:mm:ss.) Note: on units without a battery-backed real-time clock, the date and time will reset to 2000-01-01 00:00:00 upon power-up.
U	Supported in firmware version 2526+ requests the Reader to delete all stored files including data, images, and Javascript files (no *data*) (The Reader will respond with *d* or *e*.)
W	Supported in firmware version 2526+ requests the Reader to write its current settings from RAM to its non-volatile memory. (The Reader will respond with *d* or *e*.)
X	initiates data transfer from Reader memory to Host; *data* specifies whether only buffer data or all data (buffer and log) should be sent: if empty, all data are sent; otherwise, 0 requests only buffer data and 1 requests all data (buffer and log). Note: if the autoLogErase setting is nonzero, the log data are cleared after being successfully sent. (The Reader will respond with a *g* packet, zero or more *z* or *p* packets, and a final *d* or *e*. In raw mode, the *g* and *d* or *e* packets are omitted; thus if there are no data stored then no response will arrive.)
Y	acknowledge the receipt of a packet; *data* specifies the received packet number (one byte). (The Reader will not respond.)
Z	request the Reader to reboot; *data* is either empty or contains '1'; if it contains 1, the Reader will reboot into boot loader mode. (The Reader will respond with *d* or *e* before it reboots.)
\	Enhanced to include the Z1 feature in firmware version 2526+ returns a string previously stored with the *B* command; *data* contains the event number of the stored-command-code as a single byte. (The Reader will respond with a *d* packet containing the requested null-terminated string or with *e* (e.g., if the specified event number is out-of-range).)

See "Appendix: B-string Settings" for complete details.

^ requests the Reader to upload the specified stored file; *data* contains the file name, terminated with ASCII NUL.

(The Reader will respond in the same manner as with the '(' command with the following additions: in the *g* packet, *data* contains the filename followed by a tab character, followed by the file's size in parentheses (e.g., "test.txt (1292)"); in the *d* packet, *data* contains "EOF" followed by a tab character, followed by the file's CRC16 in parenthesis (e.g., "EOF (13626)").)

Supported in firmware version 3100+

_ causes the Reader to wait for all buttons to be released and clear its event queue

(The Reader will respond with *d* or *e*.)

| sends *data* to the Reader. The *data* may be intercepted by the JavaScript application on the Reader; otherwise, it will be processed as if read from a code.

(The Reader will respond with *d* or *e*.)

~ requests the Reader to write some of its current settings from RAM to its non-volatile memory. All settings are written *except* the communication settings (commMode, commProtocol, commExpectResponse, and commandOptions, uartBaud, etc.)

(The Reader will respond with *d* or *e*.)

This command is valid for Firmware versions 2230+

7 File Installation

7.1 Simple Protocol

The file is split into blocks of 236 or less bytes each and downloaded to the Reader via *1, 2, & 5* commands using the following sequence:

1) Send a *1* command to initialize the download.
2) Wait for a *d* or *e* response from the Reader or a timeout.
   a) If timeout or *e* response, restart the sequence at step 1.
   b) If *d* response, continue to step 3.
3) Send a series of *2* commands, each with a portion of the file. (The reader will not send any response.)
4) Send a *5* command to end the download and install the file.
5) Wait for a *d, e,* or *f* response from the Reader or a timeout.
   a) If *f* response or timeout, restart the sequence at step 1.
   b) If *e* response, repeat step 5.
   c) If *d* response, file download has completed successfully.

Note: the timeout will need to be increased from the normal response timeout to allow the firmware time to write the file to the flash memory.

8 Reader Settings

The Host sets the Reader settings using the */*, *O*, *P*, *Q*, and = commands and reads them using the *G*, ,, and < commands.

For example, the following P command sets register 2e to the value 7f.

```
P(2e)7f
```

Note: for two-digit setting numbers (i.e., settings 00 through fd), an alternative format may be used: in place of the parentheses and hexadecimal setting number, substitute a single character, which represents the setting number. The equivalent to the example above is `P.7f` (the ASCII '.' character has the hexadecimal value 2e). (In certain circumstances, such as with text-commands, "percent-encoding" may be used for encoding a character as a sequence consisting of the percent character followed by two hexadecimal digits. With percent-encoding, the example may be expressed as `P%2e7f`.)

In the table below, the **Reg** column is the setting number, in **hexadecimal,** to be used with the commands identified above. In the **Default** column, all values are in **hexadecimal** unless otherwise specified. To use decimal values in commands you must precede the data with a pound sign '#'. The following P command sets register 2e to the same value as the example above:

```
P(2e)#127
```

Since the single digit values of 0 through 9 are identical in decimal and hexadecimal, no indicator is needed.

8.1 Binary Dip Switch

Some registers are what Code terms a 'Binary Dip Switch' where the value of each bit of the data string switches on or off some part of the behavior of that register. The bits are numbered from least significant to most, a.k.a. right to left. Each bit can be on or off (1 or 0).

An example of this is register 0a, 'nec2of5Options'. The following settings are possible:

Bit (R to L)	Controls	Value
0	NEC2of5 Decoding	0: Disable
		1: Enable
1	Checksum checking	0: Disable
		1: Enable
2	Strip checksum from the result	0: Disable
		1: Enable
3	1 Digit Symbol Allowed	0: Disable
		1: Enable
4	2 Digit Symbol Allowed	0: Disable
		1: Enable

Given the settings above, the binary string to turn on NEC 2 of 5 decoding with checksum checking with the checksum stripped from the result string, allowing 2 digit symbols is 10111 (bits left to right). The same string would be 0x17 or decimal 23

Thus, the command to implement the settings above would be:

```
P(0a)17
```

Or

```
P(0a)#23
```

8.2 Reader Settings Table

Note: If only one default value is shown then this value is applicable and identical to both MAH200/MAH300 and MAH120. All differences in the settings use or default value will be noted.

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>00</td><td>Bluetooth Radio Out-of-range (enableRfDisconnectAlarm)</td><td>0</td><td>Binary Dip Switch
<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td rowspan="2">0</td><td rowspan="2">Vibrate</td><td>0: Disable</td></tr><tr><td>1: Enable</td></tr><tr><td rowspan="2">1</td><td rowspan="2">Beep</td><td>0: Disable</td></tr><tr><td>1: Enable</td></tr></table>Alarm when RF doesn’t have connection</td></tr>
<tr><td>01</td><td>Battery Trigger (button8)</td><td>MAH200/MAH300: 4</td><td>The event associated with the charge pin trigger. Used by the BH1, BH2, and H2.

See register 39 for a list of event numbers.

Note: Supported in firmware version 2362+</td></tr>
<tr><td>02</td><td>Battery Trigger Confirmation Time (button8ConfirmationTime_ms)</td><td>0</td><td>Valid Range: 0 to 7FFFFFFF Milliseconds

See register e3 for the list of settings.

Note: Supported in firmware version 2362+</td></tr>
<tr><td>04</td><td>Continous Illumunation Leave illumination on during read (leaveIlluminationOnDuring Read)</td><td>MAH200/MAH300: 0

MAH120: 1</td><td>0: Turn illumination off between read attempts
1: Leave illumination on between read attempts

Note: Supported in firmware version 2362+

Note: Obsolete for MAH200/MAH300 firmware 3430+</td></tr>
<tr><td>05</td><td>No Wait USB (usbKbEnumNotWaitForSet ReportLed)</td><td>0</td><td>0: Default mode, declare enumeration after receive set LED status report
1: Declare enumeration after receive get report descriptor command. Used for some Windows CE based devices

Special case for USB enumeration that doesn’t require Host keyboard response

Note: Supported in firmware version 2362+</td></tr>
</table>

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>06</td><td>Always Enable Auto White Balance (alwaysEnableAwb)</td><td>0</td><td>0: Disable
1: Enable

Auto White Balance

Note: Supported in firmware version 2378+

Note: Requires power cycle before taking effect.</td></tr>
<tr><td>07</td><td>kbSendDelay_kbClocks</td><td>0</td><td>Each keyboard clock period is 40 microseconds.

Note: Supported in firmware version 2378+</td></tr>
<tr><td>08</td><td>Reader Packet Format (commProtocol)</td><td>1</td><td>0: Reserved
1: Raw
2: Packet mode
3: Reserved</td></tr>
<tr><td>09</td><td>Keyboard Inter-Message Delay (kbInterMessageDelay_ms)</td><td>0</td><td>Valid Range: 0 to 7FFFFFFF Milliseconds

Delay between each keyboard data transmit in the unit of milliseconds

Note: Supported in firmware version 2378+</td></tr>
<tr><td>0a</td><td>NEC 2 of 5 Symbology (nec2of5Options)</td><td>1</td><td>Binary Dip Switch

<table>
<tr><th>Bit</th><th>Controls</th><th>Value</th></tr>
<tr><td rowspan="2">0</td><td rowspan="2">NEC 2 of 5 Decoding</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">Checksum checking</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
<tr><td rowspan="2">2</td><td rowspan="2">Strip checksum from the result</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
<tr><td rowspan="2">3</td><td rowspan="2">1 Digit Symbol Allowed</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
<tr><td rowspan="2">4</td><td rowspan="2">2 Digit Symbol Allowed</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
</table>

Note: All symbol lengths greater than 2 are always enabled when NEC 2 of 5 Decoding is enabled.

Note: Supported in firmware version 2378+</td></tr>
</table>

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>0b</td><td>Matrix 2 of 5 Symbology (matrix2of5Options)</td><td>1</td><td>Binary Dip Switch
<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td>0</td><td>Matrix 2 of 5 Decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>1</td><td>Checksum checking</td><td>0: Disable
1: Enable</td></tr><tr><td>2</td><td>Strip checksum from the result</td><td>0: Disable
1: Enable</td></tr><tr><td>3</td><td>1 Digit Symbol Allowed</td><td>0: Disable
1: Enable</td></tr><tr><td>4</td><td>2 Digit Symbol Allowed</td><td>0: Disable
1: Enable</td></tr></table>All symbol lengths greater than 2 are always enabled when Matrix 2 of 5 Decoding is enabled

Note: Supported in firmware version 2378+</td></tr>
<tr><td>0c</td><td>Telepen Symbology (telepenEnable)</td><td>1</td><td>0: Disable
1: Enable</td></tr>
<tr><td>0d</td><td>Enable Non-Square Data Matrix Symbology (enableNonSquareDataMatrix)</td><td>0</td><td>0: Disable
1: Enable

Supported in firmware version 2378+</td></tr>
<tr><td>0f</td><td>Targeting Control (targetEnable)</td><td>MAH200/MAH300: 3

MAH120: 3

(both bits enabled)</td><td>Binary Dip Switch
<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td>0</td><td>Green (MAH120)
Target (MAH200/MAH300)</td><td>0: Disable
1: Enable</td></tr><tr><td>1</td><td>Red (MAH120)
Target (MAH200/MAH300)</td><td>0: Disable
1: Enable</td></tr></table>Note: Setting to 0 will disable all targeting features.</td></tr>
<tr><td>14</td><td>Image transform (transformImage)</td><td>0</td><td>0: No transform
1: Mirror
2: Invert</td></tr>
<tr><td>16</td><td>DataMatrix Rectangular Symbology (dataMatrixRect)</td><td>0</td><td>0: Disable
1: Enable</td></tr>
<tr><td>19</td><td>DataMatrix Symbology (enableDataMatrix)</td><td>1</td><td>Binary Dip Switch
<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td>0</td><td>Data Matrix Decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>1</td><td>Inverse Data Matrix Decoding</td><td>0: Disable
1: Enable</td></tr></table></td></tr>
</table>

Reg	Setting Name	Default (hex)	Comment
1a	Straight 2 of 5 Symbology (enableMiscBarcodes)	1	0: Disable 1: Enable Straight 2 of 5 (with 2 or 3 start/stop codes) Decoding
1b	Communications Mode (commMode)	2	MAH200/MAH300: 0: PS/2 (AT) keyboard 1: RS232 serial 2: USB keyboard 3: Reserved 4: RF (BlueTooth) 5: USB Native (HID) 6: USB VComm (3000+ firmware) 7: USB HID POS -Terminal 131 (firmware 3484+) MAH120: 0: PS/2 (AT) keyboard (firmware 4126+ and serial number 10050561+) 1: RS232 serial 2: USB keyboard 3: Reserved 4: Not valid 5: USB Native (HID) 6: USB VComm (3000+ firmware) 7: USB HID POS -Terminal 131 (firmware 4144+) This setting is used in conjunction with settings 08 and 42 to configure the communication mode between standard “one-way” and “two-way” modes. For example, USB “two–way” native: 1b: 5 (USB Native) 08: 2 (packet mode) 42: 1 (expect response) **Note: The following must be completed within 1 second.** first output report with numlock set and capslock clear second output report with numlock set and capslock clear third output report with capslock set numlock clear fourth output report with capslock set numlock clear fifth output report with numlock set and capslock clear sixth output report with numlock set and capslock clear On the last output report comm protocol is set to raw mode, comm expect response is false and comm mode is USB Downloader mode **Note:** In firmware 3550+ register 78 also disables this feature. MAH120 enables this in 4245+ **Note:** PS/2 is not supported on the Embedded MAH120

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>1c</td><td>Baud Rate (uartBaud)</td><td>E100
(# 57600)</td><td>All standard baud rates up #115200
#9600 (2580)
#19200 (4B00)
#38400 (9600)
#57600 (E100)
#115200 (1C200)</td></tr>
<tr><td>1d</td><td>Stop Bits (uartStopBits)</td><td>2</td><td>1: Send 1 stop bit
2: Send 2 stop bits</td></tr>
<tr><td>1e</td><td>Data Bits (uartDataBits)</td><td>8</td><td>7: 7 data bits
8: 8 data bits</td></tr>
<tr><td>22</td><td>Parity (uartParity)</td><td>0</td><td>0: None
1: Odd
2: Even</td></tr>
<tr><td>26</td><td>Beep/Vibrate Volume
(beepVolume_percent)</td><td>64
(#100)</td><td>Valid Range: 0 to 64 (#100) Percent

This is the percentage of full volume.

Also see registers: 59, a7</td></tr>
<tr><td>29</td><td>PDF417 Symbology
(enablePdf417)</td><td>1</td><td>0: Disable
1: Enable

Also see registers: 2a, cf</td></tr>
<tr><td>2a</td><td>Micro PDF417 Symbology
(enableMicroPdf417)</td><td>0</td><td>0: Disable
1: Enable

Also see registers: 29, cf</td></tr>
<tr><td>2b</td><td>QR Code Symbology
(enableQrCode)</td><td>0</td><td>Binary Dip Switch

<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td>0</td><td>QR Code Decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>1</td><td>Inverse QR Code Decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>2</td><td>Micro QR Code Decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>3</td><td>Inverse Micro QR Code Decoding</td><td>0: Disable
1: Enable</td></tr></table>
Note: Micro QR is supported in firmware version 2378+</td></tr>
<tr><td>2c</td><td>Extra Time in Active Mode When Cable is Connected
(extraCabledActiveTime_ms)</td><td>6DDD00

(#120*60*1000 (7200000 ms or 2 hours))</td><td>Valid Range: 0 to 7FFFFFFF Milliseconds

Active Mode is the time between the last user interaction with the Reader (button press, etc.) and Power-Saving Idle Mode.

This setting increases Active Mode time only when the Reader is cabled.

Power Usage: 4 out of 4

Next state: Idle

Note: in “continuous read” mode, this is also the continuous read timeout period.

Also see registers: 32, 88, 8e, 9e, 9f</td></tr>
</table>

Reg	Setting Name	Default (hex)	Comment
2d	Keyboard Maps (kbMap)	0	0: US English (without leading 0 in the alt _ Num) 1: ASCII (alt+number) - universal 2: Custom (requires user to download keyboard map) 3: US English (with leading 0 in the alt + num for non-printable ASCII) 4: French Keyboard 5: German Keyboard 6: Japanese Keyboard 7: US English (with ctrl + char for non-printable ASCII) **Note:** Supported in firmware version 2394+
32	Time in Active Mode (activeTime_ms)	493E0 (#5*60*1000 (300000 ms or 5 minutes))	Valid Range: 0 to 7FFFFFFF Milliseconds. Active Mode is the time between the last user interaction with the Reader (button press, etc.) and Power-Saving Idle Mode. Power Usage: 4 out of 4 Next state: Idle **Note:** in “continuous read” mode, this is also the continuous read timeout period. Also see registers: 2c, 88, 8e, 9e, 9f
34	Maximum Candidate Decodes Per Read (maxStickyDecodes)	1	The Reader will process up to this number of codes per “read code” event. If there are more than this many codes in the field of view and within target tolerance, only the first ones will be decoded. For fastest performance with single codes, set to 1. Also see register 44
35	Button Stay-Down Time (stickyTime_ms)	0	Valid Range: 0 to 7FFFFFFF Milliseconds Keep processing the “read code” events for this amount of time (act as if the button sticks down for this time)
36	Number of Control Frames Before Picture Capture (agcFramesBeforePicture)	6	Valid Range: 0 to 7FFFFFFF Frames Number of frames captured and discarded before live picture to give the gain control time to adjust. Also see registers: 43, ab, ac, ad, ae, af, b1
37	Host Acknowledgement Time Limit (hostAckTimeout_ms)	2BC (#700)	Valid Range: 0 to 7FFFFFFF Milliseconds After sending data to host, the Reader waits up to this amount of time for the acknowledgement from host before declaring failure.

Reg	Setting Name	Default (hex)	Comment
39	Right Top Button (button1)	4	The specified event is posted upon press of this button. The events are defined below. 0: no action 1: keep awake 2: show target 3: read in near and far fields 4: defaultEvent (default) Selected by hardware - MAH200/MAH300 read in both fields - MAH120 read in near fields 5: read in near/high density field 6: read in far/wide field 7: take picture 8: read in most recently successful field 9 – 71: execute stored command string indicated by value 72 – 254: reserved 255: idle
3a	Left Top Button (button2)	4	See register 39 for settings.
3b	Combination of Left and Right Buttons (button3)	MAH200/MAH300: 0	Combination of buttons 1 and 3. See register 39 for settings.
3c	Handle Button (button4)	MAH200/MAH300: 4 MAH120:4	See register 39 for settings. **Note:** This is the register for the trigger button for the MAH120.
40	Text Command Timeout (hostPacketTimeout_ms)	2AF8 (#11000)	Valid Range: 0 to 7FFFFFFF Milliseconds The maximum time during which a complete text command from host must be received. (Pending text command data is discarded when the timeout is exceeded.)

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>41</td><td>Text Commands
(commandOptions)</td><td>8</td><td>Binary Dip Switch

<table>
<tr><th>Bit</th><th>Controls</th><th>Value</th></tr>
<tr><td>0</td><td>Text Commands</td><td>0: Disable
1: Enable</td></tr>
<tr><td>1</td><td>Suppress Echo</td><td>0: Disable
1: Enable</td></tr>
<tr><td>2</td><td>Suppress Responses</td><td>0: Disable
1: Enable</td></tr>
<tr><td>3</td><td>Disable Text Commands but Enable Magic Sequence; See Below</td><td>0: Disable
1: Enable</td></tr>
<tr><td>4</td><td>Suppresss URL Decode; See Below</td><td>0: Disable
1: Enable</td></tr>
<tr><td>5</td><td>Accept On Timeout</td><td>0: Disable
1: Enable</td></tr>
</table>

Magic Sequence:
The Magic Sequence is the string “;>PAx” where x is 1, 3, or 7 as defined above. This would normally be used in command text files, which would begin with the text-command-on sequence and end with the command to return to this special mode. For example:
;>PA7
;any desired commands here
PA8

Note: ;>PAx is supported in firmware version 2210+

Additional settings in firmware version 3430+
Suppress URL Decode:
For example, if enabled, P%418 will not equal PA8. The % is not recognized as an escape character
Accept On Timeout:
Note: See register 156 for details</td></tr>
<tr><td>42</td><td>Expect Acknowledgement From Host
(commExpectResponse)</td><td>0</td><td>0: Reader doesn’t wait for acknowledge
1: Reader will retransmit data when host doesn’t acknowledge receipt</td></tr>
<tr><td>43</td><td>JPEG Picture Quality
(jpegQuality_percent)</td><td>32
(#50)</td><td>Valid Range: 0 to 64 (#100) Percent

0: Raw Image (No JPEG Compression)
1 To 100: JPEG Compression Quality Percent

Also see registers: 36, ab, ac, ad, ae, af, b1</td></tr>
</table>

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>47</td><td>MaxiCode Symbology (enableMaxiCode)</td><td>0</td><td>Binary Dip Switch
<table>
<tr><th>Bit</th><th>Controls</th><th>Value</th></tr>
<tr><td>0</td><td>MaxiCode Decoding, Mode 0</td><td>0: Disable
1: Enable</td></tr>
<tr><td>1</td><td>MaxiCode Decoding, Mode 1</td><td>0: Disable
1: Enable</td></tr>
<tr><td>2</td><td>MaxiCode Decoding, Mode 2</td><td>0: Disable
1: Enable</td></tr>
<tr><td>3</td><td>MaxiCode Decoding, Mode 3</td><td>0: Disable
1: Enable</td></tr>
<tr><td>4</td><td>MaxiCode Decoding, Mode 4</td><td>0: Disable
1: Enable</td></tr>
<tr><td>5</td><td>MaxiCode Decoding, Mode 5</td><td>0: Disable
1: Enable</td></tr>
<tr><td>6</td><td>MaxiCode Decoding, Mode 6</td><td>0: Disable
1: Enable</td></tr>
</table></td></tr>
<tr><td>48</td><td>Codabar Checksum (miscBarcodeChecksum)</td><td>0</td><td>Binary Dip Switch
<table>
<tr><th>Bit</th><th>Controls</th><th>Value</th></tr>
<tr><td>0</td><td>Codabar Checksum Checking</td><td>0: Disable
1: Enable</td></tr>
<tr><td>1</td><td>Strip Checksum From Output</td><td>0: Disable
1: Enable</td></tr>
</table></td></tr>
<tr><td>49</td><td>Code 39 Symbology (code39FullAscii)</td><td>0</td><td>0: Disable
1: Enable

Code 39 Full ASCII Decoding</td></tr>
<tr><td>4a</td><td>Composite Codes (enableCompositeCodes)</td><td>0</td><td>0: Disable
1: Enable

Composite Code Decoding

Also see register: d8</td></tr>
</table>

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>4b</td><td>Postal Code Symbology (enablePostalCodes)</td><td>0</td><td>0: Disable
0x8: Australian Post decoding enabled
0x20000: Japan Post decoding enabled

0x200001: KIX decoding enabled
0x80: Planet decoding enabled

0x2000: Postnet decoding enabled
0x200000: Royal Mail decoding enabled
-0x80000000: USPS4CB (Firmware version 3418+)</td></tr>
<tr><td>4c</td><td>RSS Symbology (enableRss)</td><td>0</td><td>Binary Dip Switch
<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td>0</td><td>RSS Expanded decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>1</td><td>RSS Expanded Stacked decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>2</td><td>RSS Limited decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>3</td><td>RSS-14 and RSS-14 Truncated decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>4</td><td>RSS-14 Stacked and RSS-14 Stacked omni-directional decoding</td><td>0: Disable
1: Enable</td></tr></table></td></tr>
<tr><td>4d</td><td>UPC Expansion (enableUpcExpansion)</td><td>1</td><td>0: Disable
1: Enable

Also see registers: 4e, 6a, 74</td></tr>
<tr><td>4e</td><td>UPC Supplemental (enableUpcSupplemental)</td><td>1</td><td>0: Disable
1: Enable

Also see registers: 4d, 6a, 74</td></tr>
<tr><td>4f</td><td>MSI Plessey Symbology (enableMsip)</td><td>0</td><td>0: Disable
1: Enable</td></tr>
<tr><td>50</td><td>Aztec Symbology (enableAztec)</td><td>0</td><td>Binary Dip Switch
<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td>0</td><td>Aztec Decoding</td><td>0: Disable
1: Enable</td></tr><tr><td>1</td><td>Inverse Aztec Decoding</td><td>0: Disable
1: Enable</td></tr></table></td></tr>
</table>

Reg	Setting Name	Default (hex)	Comment
53	SXGA Near Field Window Vertical Size (MAH120: Horizontal) (nearFieldDecodeWindowWidth_pixels)	MAH200/MAH300: 280 (#640) MAH120: 500 (#1280)	Valid Range: 1 to 640 pixels Decoding is attempted in only the specified window of the full image. **Note:** the width and height refer to the physical image, which is rotated 90 degrees on CR2. This setting is applicable to SXGA mode only.
54	SXGA Near Field Window Horizontal Size (MAH120: Vertical) (nearFieldDecodeWindowHeight_pixels)	MAH200/MAH300: 400 (#1024) MAH120: 280 (#640)	Valid Range: 1 to 400 (#1024) Pixels Decoding is attempted in only the specified window of the full image. **Note:** the width and height refer to the physical image, which is rotated 90 degrees on CR2. This setting is applicable to SXGA mode only.
55	Notify of No-Read (notifyOfReadFailure)	0	0: Disable 1: send “r” packet on No-Read (See “r” packet in Section 5.2.) 0x100xx: post event on No-Read , where the lower 8 bits specify the event number. For example, 0x10009 to post Event 0x09. The following example will use a stored code and stored code event to output “No Read” on read failure. B% 48%01X%1d%02>No%0dRead%00 P%5510048 Output when no decode was achieved: No (%0d inserts carriage return) Read
59	Beep/Vibration Duration (beepDuration_ms)	MAH200/MAH300: 64 (#100) MAH120: 50 (#80)	Valid Range: 0 to 7FFFFFFF Milliseconds Also see registers: 26, a7
6a	UPC Symbology (enableUpc)	1	0: Disable 1: Enable Also see registers: 4d, 4e, 74
6b	Code 39 Symbology (enableCode39)	1	0: Disable 1: Enable Also see register: 70
6c	Code 93 Symbology (enableCode93)	1	0: Disable 1: Enable
6d	Code 128 Symbology (enableCode128)	1	0: Disable 1: Enable
6e	Interleave 2 of 5 Symbology (enableInterleave2of5)	1	0: Disable 1: Enable Also see registers: 71, c9
6f	Codabar Symbology (enableCodabar)	1	0: Disable 1: Enable

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>70</td><td>Code 39 CheckSum (code39Checksum)</td><td>0</td><td>Binary Dip Switch
<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td>0</td><td>Code 39 Checksum Checking</td><td>0: Disable
1: Enable</td></tr><tr><td>1</td><td>Strip Checksum From Output</td><td>0: Disable
1: Enable</td></tr></table>
Also see register: 6b</td></tr>
<tr><td>71</td><td>Interleave 2 of 5 Checksum (interleave2of5Checksum)</td><td>0</td><td>Binary Dip Switch
<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td>0</td><td>Interleave 2 of 5 Checksum Checking</td><td>0: Disable
1: Enable</td></tr><tr><td>1</td><td>Strip Checksum From Output</td><td>0: Disable
1: Enable</td></tr></table>
Also see registers: 6e, c9</td></tr>
<tr><td>72</td><td>Auto Stored Data Erase (autoLogErase)</td><td>1</td><td>0: Disable
1: Enable

Note: When “1,” data and images are cleared from nonvolatile memory when they are successfully uploaded to the host. (In “Log mode,” this is set to “0.”)</td></tr>
<tr><td>73</td><td>Auto Transfer Buffer Memory (autoBufferUpload)</td><td>1</td><td>0: Disable
1: Enable

When “1,” the reader will automatically upload buffered data (i.e., storage that hasn’t been previously uploaded) whenever a connection is present.</td></tr>
<tr><td>74</td><td>UPC Short Margin (enableUpcShortMargin)</td><td>1</td><td>0: Disable
1: Enable

Also see registers: 4d, 4e, 6a</td></tr>
<tr><td>75</td><td>RS-232 Batch Mode (uartAlwaysConnected)</td><td>0</td><td>0: Detect RS232 cable by presence of power on pin 1.
1: Assume RS232 cable is always connected</td></tr>
<tr><td>76</td><td>Storage Mode (sendAndStoreMode)</td><td>0</td><td>0: Normal mode (buffered send)
1: Send and log mode
3: Log only mode

Also see registers: 72, 73</td></tr>
<tr><td>78</td><td>Settings Lock (settingsLock)</td><td>1</td><td>1: Settings unlocked
3: Settings locked (except settingsLock)</td></tr>
</table>

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>85</td><td>Trioptic Options</td><td>1</td><td>Binary Dip Switch
<table>
<tr><th>Bit</th><th>Controls</th><th>Value</th></tr>
<tr><td rowspan="2">0</td><td rowspan="2">Trioptic Decoding, Normal Quiet Zones</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">Allow Short Quiet Zones</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
<tr><td rowspan="2">2</td><td rowspan="2">No Quiet Zones Required (requires firmware version 3280+)</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
</table></td></tr>
<tr><td>86</td><td>Motion Detection Event (motionEvent)</td><td>0</td><td>Valid Range:

0: Disable

When motion is detected, this event is posted.

See register 39 for list of events.</td></tr>
<tr><td>87</td><td>Motion Detection Sensitivity (motionThresh)</td><td>MAH200: 19 (#25)

MAH120: F (#15)</td><td>Valid Range:

The sensitivity of motion detection. Lower numbers make detection more sensitive; higher numbers, less sensitive.</td></tr>
<tr><td>88</td><td>Time in Sleep Mode (sleepTimeout_sec)</td><td>1C20 (#2*60*60 (7200 sec or 2 hours))</td><td>Valid Range: 0 to 7FFFFFFF Seconds

Sleep Mode is the time between Standby Mode and Power Off Mode. The reader can wake up out of Sleep Mode without having to reboot resulting in a quicker transition to Active Mode. Other than the quicker transition to Active Mode and slightly higher power usage, the reader is in a state very similar to Power Off.

Power usage: 1 out of 4
Communications: Disabled

Previous state: Standby
Next state: Power Off (which uses no power)

Also see registers: 2c, 32, 8e, 9e, 9f</td></tr>
<tr><td>89</td><td>Bluetooth PIN (bluetoothPin)</td><td>12345678 (String)</td><td>Value of the Bluetooth Pin sent if the host requests a pin.

Note: See register 173

Note: Supported in firmware 3514+</td></tr>
</table>

Reg	Setting Name	Default (hex)	Comment
8e	Time in Idle Mode (idleTimeout_sec)	5A (#90)	Valid Range: 0 to 7FFFFFFF Seconds Idle Mode is the time between Active Mode and Standby Mode. The reader can transition to Active Mode very rapidly. Other than the quicker transition to Active Mode and slightly higher power consumption, the reader is in a state very similar to Standby Mode. Power usage: 3 out of 4 Communications: Enabled Previous state: Active Next state: Standby Also see registers: 2c, 32, 88, 9e, 9f
93	Suppress Beep On Decode (suppressBeepOnDecode)	0	0: beep indicating decode before JavaScript processing 1: call JavaScript without beeping to indicate decode Normally, the Reader beeps as soon as decodes are read and processes them via JavaScript if necessary *after* the beep. To enable JavaScript to control the beep feedback, change this setting to 1; this will suppress the beep; the JavaScript would typically beep if the decode is valid or start another read cycle if it isn't. This setting does not suppress beeps for anything but a successful decode event. **Note:** supported in firmware version 3100+
9c	Laser Brightness (targetBrightness_percent)	64 (#100)	Valid Range: 0 to 64 (#100) Percent
9d	Target Tolerance (targetTolerance_percent)	640 (#1600)	Valid Range: 0 to 7FFFFFFF Percent For the Reader to accept a code, the target dot must be within the code rectangle or in proximity to the symbol. The nearness is defined as this percentage of the code's smaller dimension. For example, with a 10 x 20 mm code and a setting of 150 (%), the target dot must be within 15 mm of the code. Any value over 1000 is considered infinite tolerance, and no target checking is performed.
9e	Extra Time in Idle Mode When Cable is Connected (extraCabledIdleTime_sec)	7FFFFFFF (#596523*60* 60 (2147483647 sec or ~596523 hours))	Valid Range: 0 to 7FFFFFFF Seconds Idle Mode is the time between Active Mode and Standby Mode. The reader can transition to Active Mode very rapidly. Other than the quicker transition to Active Mode and slightly higher power consumption, the reader is in a state very similar to Standby Mode. Power usage: 3 out of 4 Communications: Enabled Previous state: Active Next state: Standby Also see registers: 2c, 32, 88, 8e, 9f

Reg	Setting Name	Default (hex)	Comment
9f	Time in Standby Mode (standbyTimeout_sec)	0	Valid Range: 0 to 7FFFFFFF Seconds Standby Mode is the time between Idle Mode and Sleep Mode. Standby Mode has lower power consumption but is in a state very similar to Idle Mode. Power usage: 2 out of 4 Communications: Disabled but the radio maintains its connection (RF mode) Previous state: Idle Next state: Sleep **Note:** supported in firmware version 2526+ Also see registers: 2c, 32, 88, 8e, 9e
a1	Vibrate (vibrate)	MAH200/MAH300: 0 MAH120: 1	0: Disable 1: Enable: vibrator will be on at same time as speaker **Note:** if vibrate-only is desired, set speaker volume to 0.
a2	Default Event Delay (defaultEventDelay_ms)	0	Valid Range: 0 to 7FFFFFFF Milliseconds The Reader will pause for this amount of time between each posting of the default event (used with "continuous read" mode). Also see register c4.
a7	Beep (Vibrate) Pulse Separation (beepSeparation_ms)	MAH200/MAH300: 14 (#20) MAH120: 64 (#100)	Valid Range: 0 to 7FFFFFFF Milliseconds The spacing in milliseconds between beeps. Also see registers: 26, 59
ab	AGC Selection for Taking Pictures (useImagerAgcWithTakePicture)	0	0: use decoder AGC (designed for Symbology decoding) 1: use imager AGC (optimized for pictures) Also see registers: 36, 43, ac, ad, ae, af, b1
ac	Picture Window Left Position (pictureWindowX)	0	Specify position and size of window used with "take picture." The position and size are relative to the virtual image (i.e., not the rotated physical image). **Note:** on MAH200/MAH300, overall image is 1024 x 1280. Upper half is far field; lower half is near field. Also see registers: 36, 43, ab, ad, ae, af, b1
ad	Picture Window Upper Position (pictureWindowY)	0	Also see register: ac
ae	Picture Window Width (pictureWindowWidth)	MAH200/MAH300: 400 (#1024) MAH120: 500 (#1280)	Also see register: ac

Reg	Setting Name	Default (hex)	Comment
af	Picture Window Height (pictureWindowHeight)	MAH200/MAH300: 280 (#640) MAH120: 400 (#1024)	Also see register: ac
b0	Laser Target on Before Picture (targetBeforePicture)	0	0: Laser off before picture capture 1: Laser on before picture capture Also see registers: 36, 43, ab, ac, ad, ae, af
b1	When to apply XML rules(configCodesBeforeXML)	2	0: Process configuration strings only before applying XML rules 1: Process configuration strings only after applying XML rules 2: Process configuration strings before and after applying XML rules Controls the sequence of processing configuration strings and applying XML rules.
b3	Time of retries before reader gives up sending (commMaxSendAttempts)	3	Valid Range: 1 to 7FFFFFFF **Note:** The value 1 is defined as the original send attempt but no resends. Also see register: 42 (commExpectResponse)
b4	noStoreIfNotConnected	MAH200/MAH300: 0 MAH120: 1	0: normal buffer operation 1: nothing is stored in nonvolatile memory when there is not a valid connection. If there is no active connection and data would have otherwise been stored, the Reader will indicate this fact the same as with a storage-full condition.
bf	USB Keyboard Poll Rate (keyboardPollingPeriod)	MAH200/MAH300: A (#10) MAH120: 5	Valid Range: 1 to FF (#255) Milliseconds The host is requested to poll the USB device at the specified period.
c4	Default (Continuous) Event (defaultEvent)	FF (#255)	When no button is pressed but the reader is still in active mode (i.e., not power-saving idle or sleep modes), this event will be posted. See register 39 for the list of events. The default value (idle event) disables “continuous scanning”; use one of the read events to enable “continuous scanning.”
c6	Auto Connect (autoConnectMode)	1	0: noAutoConnect (connect only on “X” and “:” commands and upload events) 1: autoConnect (attempt to establish connection when in idle mode and maintain connection when in standby mode) 2: autoReconnect (attempt to connect when there is data to send but only within specified time of last valid connection) See register ea. 3: autoConnectIfCabled (attempt to connect if reader is cabled or in charger) **Note:** This functionality is implemented in the Viewer application. If custom JavaScript is developed, caution must be taken to include this functionality.

Reg	Setting Name	Default (hex)	Comment
c7	SXGA Far Field Window Vertical Size (farFieldDecodeWindowWidth_pixels)	MAH200/MAH300: 280 (#640)	Valid Range: 1 to 280 (#640) Pixels This setting is applicable to SXGA mode only.
c8	SXGA Far Field Window Horizontal Size (farFieldDecodeWindowHeight_pixels)	MAH200/MAH300/ MAH120 not supported	Valid Range: 1 to 400 (#1024) Pixels This setting is applicable to SXGA mode only.
c9	interleave2of5Lengths	0	FFFFFFFC: 2 and 4 digit disabled FFFFFFFD: 2 digit enabled FFFFFFFE: 4 digit enabled Also see registers: 6e, 71
ca	Auto Disconnect (autoDisconnect)	0	0: retain connection until sleep or explicit disconnect command 1: disconnect from the host when there is nothing to send. (In conjunction with autoConnect (c6) and autoBufferUpload (73), the reader will connect when there is data to send, send the data, then disconnect (to allow another reader to connect to the same host).
cd	Codablock A Symbology (enableCodablockA)	0	0: Disable 1: Enable
ce	Codablock F Symbology (enableCodablockF)	0	0: Disable 1: Enable
cf	Macro PDF417 Symbology (macroPdf417)	0	0: Disable 1: Enable Also see registers: 29, 2a
d6	Ignore Duplicate Code (duplicateBlockTime_sec)	0	Valid Range: 0 (off) to 7FFFFFF Seconds Consecutive duplicate codes (i.e., codes that contain the same data) are blocked for this amount of time (in seconds). **Note: Deprecated**, but maintain for backwards compatibility. If using 3430 or greater firmware, use register 159.
d8	Compostite Linkage Control (compositeCodesRequireBothElements)	1	0: Accept any composite element 1: Only accept composite codes if both elements could be decoded. Also see register 4a.
d9	Max Connection Wait Time (connectTimeout_sec)	1E (#30)	Valid Range: 0 to 7FFFFFFF Seconds The reader will attempt connection for up to this amount of time when a connection is explicitly requested, such as when a QuickConnect code is read or an upload is requested (by event or command).
e2	imagerResolution	0	0: SXGA (1280x1024) 1: VGA (640x480) **Note:** Error beeps (6 beeps) if in a programming code.

Reg	Setting Name	Default (hex)	Comment
e3	button1ConfirmationTime_ms	0	Valid Range: 0 to 7FFFFFFF Milliseconds The button must be pressed and held for this amount of time (without change in which buttons are held down) before the button press is accepted. Setting this value greater than zero makes it easier to select combinations of buttons (e.g., button3, which is button1 and button2 pressed together).
e4	button2ConfirmationTime_ms	0	Valid Range: 0 to 7FFFFFFF Milliseconds See register e3 for the list of settings.
e5	button3ConfirmationTime_ms	0	Valid Range: 0 to 7FFFFFFF Milliseconds See register e3 for the list of settings.
e6	button4ConfirmationTime_ms	0	Valid Range: 0 to 7FFFFFFF Milliseconds See register e3 for the list of settings.
e7	button5ConfirmationTime_ms	0	Valid Range: 0 to 7FFFFFFF Milliseconds See register e3 for the list of settings.
e8	button6ConfirmationTime_ms	0	Valid Range: 0 to 7FFFFFFF Milliseconds See register e3 for the list of settings.
e9	button7ConfirmationTime_ms	0	Valid Range: 0 to 7FFFFFFF Milliseconds See register e3 for the list of settings.
ea	Reconnect Timeout (reconnectTimeout_sec)	5A (#90)	Valid Range: 0 to 7FFFFFFF Seconds See register c6 for the list of settings.
eb	Maximum reader to host packet data size (maxPacketSize)	4000 (#16384)	Valid Range: 1 to 4000 (#16384)
ec	Host Acknowledgement Time Limit Multiplier (hostAckTimeoutMultiplier_ms)	F (#15)	Valid Range: 0 to 7FFFFFFF Milliseconds When commExpectResponse (register 42) is nonzero, the Reader will wait up to hostAckTimeout_ms + dataSize * hostAckTimeoutMultiplier_ms milliseconds to receive an acknowledgement from the host.
ed	Prefix Result with AIM Symbology Identifiers (enableSymbologyIdentifierPrefix)	0	0: don't prefix with AIM identifier 1: prefix decode result with ISO/IEC standard 15424/AIM symbology identifier
f0	Allow Code 128 Short Margin (allowCode128ShortMargin)	1	0: Disable 1: Enable
f1	VGA Near Field Window Vertical Size (MAH120 Horizaontal) (vgaNearFieldDecodeWindowWidth_pixels)	MAH200/MAH300: 140 (#320) MAH120: 280 (#640)	Valid Range: MAH200/MAH300: 1 to 320 pixels MAH120: 1 to 640 pixels **Note:** Supported in firmware version 2178+
f2	VGA Near Field Window Horizontal Size (MAH120 vertical) (vgaNearFieldDecodeWindowHeight_pixels)	MAH200/MAH300: 1E0 (#480) MAH120: 1E0 (#480)	Valid Range: MAH200/MAH300: 1 to 480 pixels MAH120: 1 to 480 pixels **Note:** Supported in firmware version 2178+ **Note: Obsolete in MAH200/MAH300 version 4058+ and MAH120 4180+**

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>f3</td><td>VGA Far Field Window Vertical Size
vgaFarFieldDecodeWindow Width_pixels</td><td>MAH200/MAH300:
140
(#320)</td><td>Valid Range: 1 to 320 pixels

Note: Supported in firmware version 2178+

Note: Obsolete in MAH200/MAH300 version 4058+ and MAH120 4180+</td></tr>
<tr><td>f4</td><td>VGA Far Field Window Horizontal Size
vgaFarFieldDecodeWindow Height_pixels</td><td>MAH200/MAH300:
1E0
(#480)</td><td>Valid Range: 1 to 480 pixels
Note: Supported in firmware version 2178+

Note: Obsolete in MAH200/MAH300 version 4058+ and MAH120 4180+</td></tr>
<tr><td>f6</td><td>Code 39 Short Margin
(enableCode39ShortMargin)</td><td>1</td><td>0: disallow short margin Code 39 symbol decoding
1: allow short margin Code 39 symbol decoding

Note: Supported in firmware version 2180+</td></tr>
<tr><td>f7</td><td>Code 11 Symbology
(code11Config)</td><td>0</td><td>Binary Dip Switch

<table>
<tr><th>Bit</th><th>Controls</th><th>Value</th></tr>
<tr><td rowspan="2">0</td><td rowspan="2">Code 11 Decoding with two checksum digits checked</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">Limit to one checksum digit</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
<tr><td rowspan="2">2</td><td rowspan="2">Strip checksum(s) from the result string</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
</table>

Note: Supported in firmware version 2182+</td></tr>
<tr><td>f8</td><td>PharmaCode Symbology
(pharmaCodeConfig)</td><td>0</td><td>See Section 11.1</td></tr>
<tr><td>f9</td><td>PharmaCode Barcount
(pharmaCodeBarCount)</td><td>1004
(#4100)</td><td>Valid Range: Each 8 bits can be 04 to 10 (#16)

Bit 0 – Bit 7: min bar count, 04 to 10 (#16)
Bit 9 – Bit 15: max bar count, 04 to 10 (#16)</td></tr>
<tr><td>fa</td><td>PharmaCode Min Value
(pharmaCodeMinValue)</td><td>F
(#15)</td><td>Valid Range: F (#15) to 1FFFE (#131070)</td></tr>
<tr><td>fb</td><td>PharmaCode Max Value
(pharmaCodeMaxValue)</td><td>1FFFE
(#131070)</td><td>Valid Range: F (#15) to 1FFFE (#131070)</td></tr>
<tr><td>fc</td><td>Keep reading codes as long as button is held down
(keepReadingWhileButtonIsPressed)</td><td>0</td><td>0: Disable (requires button to be released before next scan occurs)
1: Enable

Note: when enabled, duplicateBlockTime_sec should be set to be greater than zero.</td></tr>
<tr><td>fd</td><td>log battery level and timestamp
(logBatteryLevel)</td><td>0</td><td>Valid Range: 0 (off) to FFFFFFFF

Number of scans between each log entry that includes the battery level and the timestamp</td></tr>
<tr><td>100</td><td>Backlight Timeout
(backlightTimeout_ms)</td><td>BB8
(#3000)</td><td>Valid Range: 0 to 7FFFFFFF Milliseconds

Backlight goes off automatically after this amount of time, in milliseconds, after a button press.

Note: Supported in firmware version 2526+</td></tr>
</table>

Reg	Setting Name	Default (hex)	Comment
101	Backlight Brightness (backlightBrightness_percent)	4B (#75)	Valid Range: 0 to 64 (#100) Percent Backlight is illuminated at this percent value **Note:** Supported in firmware version 2526+
102	Keypad beep volume (keypadBeepVolume_percent)	0	Valid Range: 0 to 64 (#100) Percent Beeps at specified percentage of full beeper volume whenever a keypad key is pressed **Note:** Supported in firmware version 2526+
103	Display white mode gray scale settings (lcdWhiteMode)	0000	Valid Range: (each digit): 0 to 9 For all gray scales, each digit (4 bits) divided by 9 represents the percent of time a pixel of the corresponding gray scale is turned on in each frame. The lowest 4 bits correspond to frame 1, next 4 to frame 2, etc. **Note:** Supported in firmware version 2526+
104	Display light gray mode gray scale settings (lcdLightGrayMode)	0097	Valid Range: (each digit): 0 to 9 For all gray scales, each digit (4 bits) divided by 9 represents the percent of time a pixel of the corresponding gray scale is turned on in each frame. The lowest 4 bits correspond to frame 1, next 4 to frame 2, etc. **Note:** Supported in firmware version 2526+
105	Display dark gray mode gray scale settings (lcdDarkGrayMode)	9996	Valid Range: (each digit): 0 to 9 For all gray scales, each digit (4 bits) divided by 9 represents the percent of time a pixel of the corresponding gray scale is turned on in each frame. The lowest 4 bits correspond to frame 1, next 4 to frame 2, etc. **Note:** Supported in firmware version 2526+
106	Display black mode gray scale settings (lcdBlackMode)	9999	Valid Range: (each digit): 0 to 9 For all gray scales, each digit (4 bits) divided by 9 represents the percent of time a pixel of the corresponding gray scale is turned on in each frame. The lowest 4 bits correspond to frame 1, next 4 to frame 2, etc. **Note:** Supported in firmware version 2526+
10b	Enable JavaScript (enableJavaScript)	1	0: Disable 1: Enable When set to 0 installed scripts are disabled. This can be useful from boot mode for recovering the unit if a non-responsive script is installed. **Note:** supported in firmware version 2526+

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>10c</td><td>RF Connected Cache Time (rfConnectedCacheTime_sec)</td><td>3</td><td>Valid Range: 0 to 7FFFFFFF Seconds

The time the last connection status received from the radio remains valid.
If a request is made during this time since last radio query, the cached status is returned. Otherwise, the Reader will query the radio for connection status (which takes up to 1 second).</td></tr>
<tr><td>10d</td><td>DataMatrix Symbol Size (dataMatrixSymbolSize)</td><td>MAH200/MAH300: 0

MAH120: 2</td><td>0: Normal effort (Default)
1: Increase effort
2: Max effort

Increases the decoder’s effort to find a DataMatrix symbol in an image.

Note: supported in firmware version 3100+

Also see register 1b2</td></tr>
<tr><td>10e</td><td>Legacy Picture Upload (legacyPictureUpload)</td><td>1</td><td>Selection of picture upload method:
0: Store pictures as files – Pictures will be stored as files and must be uploaded using the “^” command. Take-picture event will store rather than immediately upload picture.
1: Legacy picture upload – Take-picture event will attempt to immediately upload picture using the legacy image upload protocol. (If upload fails, the picture will be stored as a file.).

Also, the ‘X’ command will cause stored picture files to be uploaded using the legacy image upload protocol. (The images will not be automatically transferred when a connection is established; the ‘X’ command is needed.)

Note: supported in firmware version 3100+</td></tr>
<tr><td>12c</td><td>Data Matrix Improvement (dataMatrixMiscImprovement)</td><td>0</td><td>Binary Dip Switch

<table>
<tr><th>Bit</th><th>Controls</th><th>Value</th></tr>
<tr><td rowspan="2">0</td><td rowspan="2">Binarization Improvement</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">Low Contrast Improvement</td><td>0: Disable</td></tr>
<tr><td>1: Enable</td></tr>
</table>

Improves the decoding capability of the reader on low contrast or pixilated Data Matrix barcodes

Note: Supported in firmware version 3280+</td></tr>
</table>

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>12d</td><td>Hong Kong 2 of 5 Symbology</td><td>0</td><td>Binary Dip Switch
<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td rowspan="2">0</td><td rowspan="2">Hong Kong 2 of 5 Decoding</td><td>0: Disable</td></tr><tr><td>1: Enable</td></tr><tr><td rowspan="2">1</td><td rowspan="2">1 Digit Symbol Allowed</td><td>0: Disable</td></tr><tr><td>1: Enable</td></tr><tr><td rowspan="2">2</td><td rowspan="2">2 Digit Symbol Allowed</td><td>0: Disable</td></tr><tr><td>1: Enable</td></tr></table>
Note: Supported in firmware version 3280+</td></tr>
<tr><td>12f</td><td>Notify Of Packet Rejection (notifyOfPacketRejection)</td><td>1</td><td>0: Disable
1: Beep 3 times
0x100xx: post event on No-Read , where the lower 8 bits specify the event number. For example, 0x10009 to post Event 0x09.

Specify the behavior when a packet is rejected because of incorrect encryption key, incorrect packet protocol, or XML Modem locked to a different reader.

Note: Supported in firmware version 3280+</td></tr>
<tr><td>137</td><td>PDF417 Handle Invalid Shift (pdfHandleInvalidShift)</td><td>0</td><td>0: Disable
1: Enable

Allows the decoding of PDF417 barcodes that were improperly encoded

Note: Supported in firmware version 3280+</td></tr>
<tr><td>150</td><td>Background Bluetooth Connection (synchronousSend)</td><td>0</td><td>0: Bluetooth connection is confirmed before user is allowed to scan. Pressing the trigger button will cancel the connection.
1: Allows user to begin scanning before the Bluetooth connection is confirmed.

Allows user to begin scanning before Bluetooth connection is confirmed

Note: supported in firmware version 3430+

Also see registers: 151, a0</td></tr>
<tr><td>151</td><td>Beep before Bluetooth Connection is established (quickConnectNotWaitForConnection</td><td>0</td><td>0: Beep upon establishing a Bluetooth Connection
1: Do not wait for connection before beep.

Give the second beep before connection is established when scanning QuickConnect code. To be used with register 150 to improve user experience with QuickConnect code

Note: supported in firmware version 3430+

Also see register: 150</td></tr>
</table>

<table>
<tr><th>Reg</th><th>Setting Name</th><th>Default (hex)</th><th>Comment</th></tr>
<tr><td>154</td><td>Enable Black and White Pictures (takeBlackAndWhitePicture)</td><td>0</td><td>0: Images remain in grey scale.
1: Captured images are converted to black and white.

Converts grey scale images to black and white

Note: supported in firmware version 3430+</td></tr>
<tr><td>159</td><td>Ignore Duplicate Code (duplicateBlockTime_msec)</td><td>0</td><td>Valid Range: 0 (off) to 7FFFFFF Milliseconds

Consecutive duplicate codes (i.e., codes that contain the same data) are blocked for this amount of time (in milliseconds).

The actual block time is the sum of settings d6*1000+159.

Note: Supported in firmware MAH120 version 4112+, CR2/CR2 version 3430+

Also see register: d6</td></tr>
<tr><td>172</td><td>Automatically Save Active RF Connection Bluetooth Address (rfAutoSaveActiveConnect)</td><td>MAH200/MAH300:
0</td><td>0: Disable
1: Enable

Automatically save the RF connection address. The address in the QuickConnect code will be saved if enabled.

Other communication settings such as RF one-way or RF two-way are not automatically saved, so the reader must be saved in the proper communication mode for this feature to work properly.

Note: Supported in firmware version 3474+</td></tr>
<tr><td>173</td><td>Enable Bluetooth Encryption (enableBluetoothEncrytion)</td><td>0</td><td>0: Disable
1: Enable

Enables the standard AES 128 bit Bluetooth encryption.

Note: Supported in firmware 3514+

Also see register 89</td></tr>
<tr><td>175</td><td>Rotate CR3 display and keypad (rotateKeypadDisplay)</td><td>0</td><td>Binary Dip Switch

<table><tr><th>Bit</th><th>Controls</th><th>Value</th></tr><tr><td>0</td><td>Screen Rotation 180°</td><td>0: Disable
1: Enable</td></tr><tr><td>1</td><td>Swap Left and Right Arrow Keys</td><td>0: Disable
1: Enable</td></tr></table>
Note: Requires reboot to take effect
Note: Supported in firmware 3546+</td></tr>
<tr><td>17f</td><td>jsMaxMemory_bytes</td><td>MAH200/MAH300:
600000
(6*1024*1024 or #6291456)

MAH120: C00000
(12*1024*1024 or #12582912)</td><td>JavaScript Maximum Memory Usage</td></tr>
</table>

Reg	Setting Name	Default (hex)	Comment
180	jsPeakMemory_bytes	0	Readonly. JavaScript Peak Memory Usage
1b2	HD DataMatrix Symbol Size (hdf_dataMatrixSymbolSize)	MAH200/MAH300/ MAH120 not supported	0: Normal effort (Default) 1: Increase effort 2: Max effort Increases the decoder's effort to find a DataMatrix symbol in an image – HD field only. **Note:** supported in firmware version 4008+

9 Radio Commands

The Host controls the radio by issuing *':'* commands. The following table describes the available commands.

The '#' column is the radio setting/command number (in hexadecimal) to be used with the *':'* command. For example, ":%0E" gets the Bluetooth device address.

The '# bytes' column indicates how many bytes of data are required as arguments for the command.

Name	**#**	**Comments**	**# bytes**
Disconnect	00	Terminate the current connection.	0
Auto Connect	07	If connection information exists for the specified Device Address, use it to establish a connection. Otherwise, attempt to establish a connection and store the resulting information.	6
Clear Setup	08	Remove connection information associated with the specified BlueTooth Address.	6
Send Setup	09	Print all connection information in the following format iiii xxxxxxxxxxxx p Where iiii is the storage index, xxxxxxxxxxxx is the BlueTooth Device Address, and p indicates pairing enabled (y) or pairing disabled (n).	0
Get Bluetooth Address	0E	Get Bluetooth address as 12 Hexdecimal characters	0
Get "user friendly" name	0F	Get device's "user friendly" name	0

10 Code Reader Batch (CRB) System

The Code Reader Batch (CRB) system is a convenient method for creating and maintaining a set of commands that can be easily sent to the reader. These CRB files can be created in any text editor with the file extension of .crb. The CRB system accepts all of the valid *text commands*. The most commonly used commands are *J*, *N*, *P*, and ~. There should be one command per line. The CRB file may contain empty lines and comments as well.

The .crb files can be sent to the reader using either the USB or RS232 downloader's. As CRB files are just a list of *text commands*, they can also be sent by a serial terminal program. **Note: if using a serial terminal program the reader will first need to be commanded in to "text command mode"; see Section 6.1.**

You can request a copy of all the CR2 Users' Manual configuration codes in the .crb format. For example, code M121_01 in the manual (setting the Bluetooth radio timeout to 5 minutes) is the following:

File M121_01.crb:

```
P%9f#300
```

The CRB system allows for combining multiple functions into a single file. For example, you can create a test.crb file that contains commands to set the left trigger to B3 (M030_01), enable ALL RSS codes (M267_01), disable auto-transfer of buffer memory (M069_01), and make the reader beep three times. Note you can also comment your CRB file with a semicolon (;) as shown below. A comment starts with a semicolon character and lasts till the end of the line. The ***;>PAx*** sequence has special meaning. See Section 6.1.

File test.crb:

```
; This is a test file
;>PA7          ; enable text commands, no echo/responses (see Section 8, reg 41)
P%3A13         ; left trigger set to B3 performance (note that value 13 is in hex)
P%4c#31        ; enable ALL RSS codes (note that value 31 is in decimal)
P%730          ; disable Auto Transfer Buffer Memory
; beep reader three times using a different type of sound
P%59#30        ; set Beep/Vibration Duration to 30 milliseconds
P%26#60        ; set Beep/Vibrate Volume to 60%
#%03           ; beep reader 3 times
P%59#100       ; restore Beep/Vibration Duration to 100 milliseconds
P%26#100       ; restore Beep/Vibrate Volume to 100%
P(100)#6000    ; sets the backlight timeout to 6 seconds (6s is in decimal)
P(101)55       ; dimmers setting (LCD screed) to 85% (55 hex)
~              ; save settings (except communication-related settings)
PA8            ; turn off text commands (to avoid inadvertent commanding)
```

Non-printable ASCII (0x00-0x1F and 0x7F) and non-ASCII (0x80-0xFF) characters should be encoded. See Section 6.1. Also, the following characters have special meaning in a CRB file, thus they should be encoded if they are part of a command:

0x20 ' ' (space): Space and tab characters mark the end of a command.

0x3B ';' (semicolon): Semicolon characters mark the beginning of a comment.

11 Symbology Detail Settings

11.1 PharamaCode

PharmaCode setting register (f8) contains a number of settings that requires detailed explanation. Below is a list of valid register settings and detailed explanation.

0 = disable PharmaCode decoding (Default)

1 = enable PharmaCode decoding, no color bars expected; standard rules for all bars. Horizontally oriented symbols are decoded. Note that horizontally oriented means that the bars are perpendicular to the orientation of the pixel raster scan line. Decoding is performed in the “normal” direction (left bars more significant than right bars for horizontal symbols; top bars more significant than bottom bars for vertical symbols).

3 = enable PharmaCode decoding, Color bars expected; relaxed contrast rules for the three least significant bars. Horizontally oriented symbols are decoded. Note that horizontally oriented means that the bars are perpendicular to the orientation of the pixel raster scan line. Decoding is performed in the “normal” direction (left bars more significant than right bars for horizontal symbols; top bars more significant than bottom bars for vertical symbols).

5 = enable PharmaCode decoding, no color bars expected; standard rules for all bars. Vertically oriented symbols are decoded. Note that vertically oriented means that the bars are parallel to the orientation of the pixel raster scan line. Decoding is performed in the “normal” direction (left bars more significant than right bars for horizontal symbols; top bars more significant than bottom bars for vertical symbols).

7 = enable PharmaCode decoding, Color bars expected; relaxed contrast rules for the three least significant bars. Vertically oriented symbols are decoded. Note that vertically oriented means that the bars are parallel to the orientation of the pixel raster scan line. Decoding is performed in the “normal” direction (left bars more significant than right bars for horizontal symbols; top bars more significant than bottom bars for vertical symbols).

9 = enable PharmaCode decoding, no color bars expected; standard rules for all bars. Horizontally oriented symbols are decoded. Note that horizontally oriented means that the bars are perpendicular to the orientation of the pixel raster scan line. Decoding is performed in the “reverse” direction (right bars more significant than left bars for horizontal symbols; bottom bars more significant than top bars for vertical symbols).

11 = enable PharmaCode decoding, Color bars expected; relaxed contrast rules for the three least significant bars. Horizontally oriented symbols are decoded. Note that horizontally oriented means that the bars are perpendicular to the orientation of the pixel raster scan line. Decoding is performed in the “reverse” direction (right bars more significant than left bars for horizontal symbols; bottom bars more significant than top bars for vertical symbols).

13 = enable PharmaCode decoding, no color bars expected; standard rules for all bars. Vertically oriented symbols are decoded. Note that vertically oriented means that the bars are parallel to the orientation of the pixel raster scan line. Decoding is performed in the “reverse” direction (right bars more significant than left bars for horizontal symbols; bottom bars more significant than top bars for vertical symbols).

15 = enable PharmaCode decoding, Color bars expected; relaxed contrast rules for the three least significant bars. Vertically oriented symbols are decoded. Note that vertically orientesd means that the bars are parallel to the orientation of the pixel raster scan line. Decoding is performed in the “reverse” direction (right bars more significant than left bars for horizontal symbols; bottom bars more significant than top bars for vertical symbols).

12 OCR Template

12.1 OCR Introduction

OCR is designed to encompass both human readable and machine readable information in the same symbol or text. Conversely, barcodes were designed to greatly assist the ability of machines to read information at the expense of human readability. In OCR, there is little redundant information in a character; most of an OCR character must be present to allow recognition. There are subtle differences between some OCR characters that are easy for a human to distinguish, but present challenges to machine vision systems in the presence of lower sample density, noisy images, and/or degraded symbols.

While much effort has been spent to provide a superior OCR capability, users should be cautioned that OCR decoders are more susceptible to misreads and noreads than their barcode counterparts. Consequently, The OCR decoder reads OCR only when it is provided with templates detailing the specifics of the text to read. This allows the software to distinguish the text of interest from random text that may be present in the same image. In addition, the OCR decoder supports a checksum capability to reduce the probability of misreads.

Multiple templates may be active at the same time for more user flexibility.

12.2 OCR Overview

The following OCR characters are currently supported:

OCR-A:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
#$&()*+-./<>@\€£¥

OCR-B:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
#$&()*+-./<>@\€£¥

The templates define the OCR font as well as the layout of the text in a row, column format. Each row can have up to 50 chars, with up to 18 rows in a template. However, the total number of characters can not exceed 320 characters. Within each character position, the allowable characters can be specified either through explicit ASCII values, groups of ASCII values, wildcard characters, or combinations of these types. To achieve better OCR results, it is desirable to limit the values that a character position can take to the known values that will occur in an application.

The OCR decoder can also handle spaces within OCR text with some restrictions. Internal gaps longer than one space are not allowed in templates. For example, the OCR-A text

ONE SPACE

is valid because there is only one space between the E and S in the text. However, the following text is illegal given the two spaces between the O and S:

TWO SPACES

The OCR decoder can handle arbitrary number of spaces at the beginning and end of a line. These spaces must be explicitly included in the template with the ASCII value of a space (32) and not be included as part of a group or wildcard character.

The OCR decoder also provides a checksum capability to reduce the probability of misreads. There are two types of checksums provided: row and block. A row checksum provides a checksum for all characters from the checksum to the first character of a single row in a template. A block checksum provides a checksum from its character position all the way to the first character of the overall template. For additional checksum protection, four different weighting schemes are supported: 1, 12, 13, and 137. Finally, the checksum calculation is based on modulo arithmetic. The modulo factor may vary from 6 to 36.

The OCR decoder is designed to read OCR text that is within a certain sampled range in pixels. The ideal height of an OCR character after sampling is about 20 pixels. The OCR decoder will read characters that are up to 50 pixels in height. If OCR characters are consistently above 40 pixels in height, downsampling the image by a factor of 2 before submitting it to OCR decoder will achieve better results in both speed and decode rates.

12.3 OCR Output String Values

There is no 7 bit ASCII representation for the Euro, Pound, or Yen currency characters. The 8 bit values that are returned in the result string for these characters are based on the typical code page values used in Windows. The 8 bit codes used are as follows:

Character Value

€ - Euro 128
£ - Pound 163
¥ - Yen 165

12.4 OCR User Templates

User Templates are NULL terminated strings made up of various control codes along with standard ASCII values. The control codes are assigned to ASCII values below the value of 32. The currently defined control codes are:

Control Code Definition	Value	Argument
End of Template	0	
New Template	1	Font [1-3]
New Line	2	
Define Group Start	3	ID [1-255]
Define Group End	4	
Wildcard: Numeric (0-9)	5	
Wildcard: Alpha (A-Z)	6	
Wildcard: Alphanumeric	7	
Wildcard: Any (including space)	8	
Defined Group	10	ID [1-255]
In Line Group Start	11	
In Line Group End	12	
Checksum	13	Weights, Type, MOD

12.4.1 End of Template (0)

All templates end with the ***End of Template*** control code.

12.4.2 New Template (1)

A user template may contain multiple distinct templates all within the same string. These distinct templates all begin with the ***New Template*** control code. The value immediately following this control code indicates the font(s) for which this template is designed. The current valid font values are

Font Value Active Fonts:

1 OCR-A
2 OCR-B
3 OCR-A and OCR-B

As an example, the following byte sequence reads 8 alphanumeric digits in either OCR-A or OCR-B and is the default user template:

1,3,7,7,7,7,7,7,7,0

12.4.3 New Line (2)

The OCR decoder supports mult-line templates. A new line within a multiline template is indicated by the ***New Line*** control code.

12.4.4 Define Group Start (3)

In a given character position, the user specifies which values a text character may take. To reduce the overall size of templates, users may define common groups of ASCII characters and then use the defined group rather than repeating the same sequence over and over. Groups can be made up of individual ASCII values or wildcard values. The wildcard values are control codes ***Numeric*** (5), ***Alpha*** (6), ***Alphanumeric*** (7), and ***Any***(8). Groups may **not** be nested.

To define a group, specify the ***Group Start*** control code followed by a single byte ***ID*** value that may range from 1 to 255. Up to 255 groups may be defined in a single template. Once a group is defined, you may not define another group with the same group ***ID***. Following its definition, a group may be used in any subsequent individual template.

For example, say we want to read an 8 character OCR-B text where each character may be a hexadecimal digit (0123456789ABCDEF). We can define a group that would begin the definition with the ***Group Start*** control code (3), followed by its ID (1), followed by the ***Numeric*** wildcard (5), followed by the ASCII values of the six desired letters. Finally all group definitions are terminated with the ***Group End*** control code (4):

1,2,**3**,**1,5,65,66,67,68,69,70**,**4**,10,1,10,1,10,1,10,1,10,1,10,1,10,1,10,1,0

The sequence of numbers of 65 through 70 are the decimal ASCII values for the upper case letters A through F. To use the defined group, we use the ***Defined Group*** (10) control code followed by the group ID. Each sequence of 10,1 above occupies a single character position in the OCR text to be read. The remainder of the example above include the ***Template Start*** (1) control code at the

beginning followed by the OCR-B font designator (2). Finally, the template is ended with the ***End of Template*** (0) control code.

12.4.5 Define Group End (4)

The ***Define Group End*** control code is used to terminate a ***Defined Group***. A ***Define Group End*** must always be preceded by a ***Define Group Start*** and conversely, a ***Define Group Start*** must always be followed by a ***Define Group End***.

12.4.6 Wildcard: Numeric (5)

The ***Numeric Wildcard*** control code may be used anywhere a single ASCII character code may be used and indicates that the current text character can be any of the 10 numeric digits.

12.4.7 Wildcard: Alpha (6)

The ***Alpha Wildcard*** control code may be used anywhere a single ASCII character code may be used and indicates that the current text character can be any of the 26 upper case alphabetic letters.

12.4.8 Wildcard: Alphanumeric (7)

The ***Alphanumeric Wildcard*** control code may be used anywhere a single ASCII character code may be used and indicates that the current text character can be any of the 26 upper case alphabetic letters or any of the 10 numeric digits.

12.4.9 Wildcard: Any (8)

The ***Any Wildcard*** control code may be used anywhere a single ASCII character code may be used and indicates that the current text character can be any valid character that the designated font for this template supports. For OCR-A and OCR-B this includes the 26 upper case letters, the 10 numeric digits, and the 16 punctuation characters. It includes the space character as well.

12.4.10 Defined Group (10)

The ***Defined Group*** control code uses a previously defined group. The byte immediately following the ***Defined Group*** control code must match a previously defined group. This group occupies one character position in the template.

12.4.11 In Line Group Start (11)

The ***In Line Group Start*** defines a one time instance of a group that occupies one character position in the template. Use this control code for unique groups of characters that occur only once. One could use a ***Define Group Start*** control code instead. The inclusion of both types is done for the convenience of the user.

The ***In Line Group*** must always be ended with an ***In Line Group End*** control code.

12.4.12 In Line Group End (12)

The ***In Line Group End*** control code is used to terminate an active ***In Line Group*** definition.

12.4.13 Checksum (13)

Checksums may be used to reduce the probability of misreads involving OCR. The OCR decoder supports a number of options associated with checksums. The user may specify the type (block or row), the weight scheme (1, 12, 13, 137) and the modulo value of the checksum (6-36). The byte immediately following the ***Checksum*** control code defines the type of checksum that will be used:

Bit Position(s)	Meaning
7,6: Weight Scheme	00: Weight Scheme: 1
	01: Weight Scheme: 12
	10: Weight Scheme: 13
	11: Weight Scheme: 137
5: Checksum Type	0: Row
	1: Block
4-0: Modulo Value	Checksum Modulo - 5

Row Checksums perform a checksum calculation on all characters preceding them up to the first char on the same row. Block Checksums perform a checksum calculation on all characters up to the very first character in the template; they span multiple rows.

The 5 bit Modulo Value stores the ***Checksum Modulo*** – 5. The stored number may range from 1, which is ***a Checksum Modulo*** value of 6, to 31, which describes a ***Checksum Modulo*** of 36. A Modulo value of 0 (***Checksum Modulo*** of 5) is illegal.

The characters within a checksum field have a numerical value that is used in the checksum calculation.

Digits are converted to their numerical value (0-9), while Upper case letters range from 10 for an 'A' up to 36 for a 'Z'. All punctuation characters currently have a value of 0 for checksum purposes. However, they do count as a spot for determining the weight values that will be used in calculating the checksum.

The ***Weight Scheme*** defines how the values described above can be changed based on their character position. The default weight scheme is 1. This means that the checksum is based only on the character value and is not dependent on its position. The other weight schemes multiply the character value by a repetitive weight value that helps in identifying characters that have had their column locations switched. The 4 weight schemes are:

Weight Scheme Multiplier values

1	1,1,1,1,1,….
12	1,2,1,2,1,2,….
13	1,3,1,3,1,3,….
137	1,3,7,1,3,7,1,3,7,….

The checksum character itself always start with a weight of 1. As we move away from the checksum towards the left, we update the weight value to the next member of the sequence. The sequences repeat over and over until the first character in a row for a ***Row*** type checksum, and to the first character in the template for a ***Block*** type checksum. The resulting sum is then divided by the ***Checksum Modulo*** number of the checksum and the remainder of this division should be zero for a valid checksum.

For example, the following 2 line OCR-B template contains a mod 10 checksum with weight 1 (5) at the end of each line along with a mod 36 block checksum with weight 13 (191) as the last character. There are a total of 8 alphanumeric characters per line including the checksums:

1,2,7,7,7,7,7,7,7,**13,5**,2,7,7,7,7,7,7,7,**13,5,13,191**,0

12.4.14 Example OCR User Templates

This section gives some examples of valid OCR User Templates along with the OCR text they are designed to read.

12.4.14.1 Multi_Row with Leading and Trailing Spaces

123456
** ABCDEFG**

The OCR-B text above is made up of two lines, the top being purely numeric and the bottom purely alphabetic. Also, the second line is offset from the first by two spaces. The following template will read this text:

1,2,5,5,5,5,5,5,2,32,32,6,6,6,6,6,6,6,0

Note the 2 following the 6 numeric wild card digits: this indicates that the template is inserting a new line. Also of note are the leading two spaces on the second line. They must be explicitly indicated by using the ASCII value of a space: decimal value 32. You may not use a wildcard or group to indicate leading (or trailing) space. Finally, the trailing spaces on line 1 do not need to be explicitly entered into the template. They are assumed to be there based on the number of character positions defined for the row.

12.4.14.2 Row and Block Checksums

ABCD6
EFG5X

The two lines of OCR-B alphabetic text above both contain a row checksum. In addition, the last character of row 2 is a block checksum. The 2 row checksums are mod 10 with a 13 weight (control code 133), while the block checksum is a mod 36 with a 137 weight (control code 255). The following template will read this text:

1,2,6,6,6,6,13,133,2,6,6,6,13,133,13,255,0

The top line checksum is the 6 at the end of the line. While this example shows the checksum at the end of the line, it may appear anywhere on the line and then protects all the characters to its left. The following sum is generated to verify a proper checksum on line 1:

```
‘6’ ‘D’ ‘C’ ‘B’ ‘A’
(1x6)+(3x13)+(1x12)+(3x11)+(1x10) = 100
```

Note the 13 weight scheme starting with a 1 on the checksum digit, and then alternating between a 1 and 3 for all digits to the left of the checksum up to the first character on the line. The numerical values of the alphabetic characters range from 10 for an ‘A’ up to a 36 for a ‘Z’. The sum of 100 is a multiple of 10, so the mod 10 checksum here has passed.

On line 2, the row checksum is the 5 following the G. Verifying its line by generating its sum:

```
’5’ ‘G’ ‘F’ ‘E’
(1x5)+(3x16)+(1x15)+(3x14) = 110
```

Again, we have obtained a value that is a multiple of 10, validating this row checksum.

The X at the end of the line is a mod 36 block checksum with 137 weighting. It protects all the characters in the template, including the first line. Calculating its sum working backwards from the block checksum and using the 137 weighting scheme:

```
'X' '5' 'G' 'F' 'E' '6' 'D' 'C' 'B' 'A'
(1x34)+(3x5)+(7x16)+(1x15)+(3x14)+(7x6)+(1x13)+(3x12)+(7x11)+(
1x10) = 396
```

The resulting sum is a multiple of 36, so the block checksum has been validated.

12.4.14.3 Multiple Individual Templates

A single template may contain multiple individual templates. For example consider the following two distinct OCR text strings that are to be read with the same template:

A1234 B5678

The first string is in OCR-A font while the second is OCR-B. Assume that the 'A' and 'B' are always present as the first character of the two strings and that the 4 numeric characters may be any digits. A user may code these up as two distinct templates with the following:

1,1,65,5,5,5,5,1,2,66,5,5,5,5,0

Notice the multiple use of the ***New Template*** control code (1) to separate the two individual templates along with the ***Font ID*** of 1 for the first OCR-A string and the 2 for the OCR-B string. In addition, the decimal values of 65 and 66 are the ASCII codes of the upper case letters A and B, respectively, and indicate that those character positions will always have those values.

12.5 OCR Internal Templates

The OCR decoder currently supports three internal templates: Passport, ISBN, and Price Field.

12.5.1 Passport Internal Template

This template may be used to read Passports based on the ICAO standard. There are two rows of 44 OCRB characters that make up the machine readable portion of a Passport. Each row is read separately and only one row may be issued if there is a problem decoding one of the rows. If there are multiple Passports in the image, The OCR decoder will attempt to read all of the Passports, but consecutive reads are not necessarily associated with the same Passport. Note that when the Passport Internal Template is enabled, all other templates **MUST** be disabled. The OCR decoder verifies all the various checksums that are built within the lower machine readable line. The output string provided by The OCR decoder is exactly as it is printed on the Passport page. Parsing the data to extract the various fields within it is the responsibility of the user.

The following is an example of OCR-B text of a valid Passport:

P<UTOERIKSSON<<ANNA<MARIA<<<<<<<<<<<<<<<<<<<
L898902C<3UTO6908061F9406236ZE184226B<<<<<14

One factor complicating the reading of Passport data is the varied backgrounds that exist behind the OCR data. For example, at the time this document was issued, the United States Passport has a variety of colored background patterns that can interfere with OCR decoding when the passport is illuminated with visible light. However, if the US Passport is illuminated with Infra-Red (IR) illumination, the background patterns are reduced and contrast with the OCR text is enhanced. In the

absence of IR illumination, over exposure of the target such that the background is saturated while keeping the text black can help.

12.5.2 ISBN Internal Template

This template is used to read the International Standard Book Number (ISBN) number in machine readable form. It will read the ISBN in either OCR-A or OCR-B fonts. The current ISBN format is as follows:

ISBN 0-8436-1072-7

It consists of the 4 letters ISBN followed by 13 characters that are made up of hyphen separated fields. The hyphen placement varies depending on the numerical value of the encoded data. The last digit is a Mod 11 checksum which can be any of the 10 numerical digits or an ‘X’. All ISBN issued results are checked for a valid checksum. The space between the N in the ISBN string may not be present. The OCR decoder supports either format. Also, according to the ISBN specification, the hyphens in the ISBN string may be replaced by spaces. The OCR decoder supports this format as well.

There is a new ISBN format that will be in effect as of January 1, 2007. It adds an additional 4 characters to the original ISBN number:

ISBN 978-0-571-08989-5

The ISBN template supports this format along with the original. All the comments associated with the original ISBN format discussed above apply here as well. The one exception it the checksum is now a Mod 10 checksum and as a result, the checksum can now only be in the range from 0 – 9 and does not take the value ‘X’ anymore.

12.5.3 Price Field Internal Template

The Price Field is used in a number of applications including book pricing. The format of the field is as follows:

C1234 P5678E

The field begins with a ‘C’ and ends with an ‘E’. The first part of the Price Field is a ‘C’ followed by four numeric digits. The second half begins with a currency character. The above example shows the letter ‘P’ but the Price Field template allows the following additional characters:

$€£¥

Following the currency character, a numeric grouping of 3, 4, 5 or 6 digits is followed by a terminating letter ‘E’. The template supports both OCR-A and OCR-B fonts.

The following examples will also be recognized by the The OCR decoder Price Field internal template:

C6712 $801E
C0217 €4399E
C0823 ¥31559E
C0331 £706213E

13 Appendix: B-String Settings

B%0d%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa120%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02$%03

B%0e%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#832%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa100%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02$%03

B%0f%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#480%04%01X%1d%02P%53#480%04%01X%1d%02P%c8#640%04%01X%1d%02P%c7#480%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa60%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02$%03

B%11%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa90%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02$%03

B%12%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#640%04%01X%1d%02P%53#512%04%01X%1d%02P%c8#832%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa75%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02$%03

B%13%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%f2#480%04%01X%1d%02P%f1#320%04%01X%1d%02P%f4#480%04%01X%1d%02P%f3#320%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa50%04%01X%1d%02P%dc#90%04%01X%1d%02P%e20%04%01X%1d%02$%03

B%14%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa90%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02$%05

B%15%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#640%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa80%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02$%05

B%16%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%f2#480%04%01X%1d%02P%f1#320%04%01X%1d%02P%f4#480%04%01X%1d%02P%f3#320%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#10%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa50%04%01X%1d%02P%dc#90%04%01X%1d%02P%e20%04%01X%1d%02$%05

B%17%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#1024%04%01X%1d%02P%c7#640%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#30%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa90%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02$%06

B%18%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%54#1024%04%01X%1d%02P%53#640%04%01X%1d%02P%c8#832%04%01X%1d%02P%c7#512%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#

30%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa80%04%01X%1d%02P%dc#256%04%01X%1d%02P%e20%04%01X%1d%02$%06

B%19%01X%1d%02P%640%04%01X%1d%02P%920%04%01X%1d%02P%951%04%01X%1d%02P%a3177%04%01X%1d%02P%a5177%04%01X%1d%02P%b5177%04%01X%1d%02P%b60%04%01X%1d%02P%28122)0%04%01X%1d%02P%f2#480%04%01X%1d%02P%f1#320%04%01X%1d%02P%f4#480%04%01X%1d%02P%f3#320%04%01X%1d%02P%5c1%04%01X%1d%02P%5d#12%04%01X%1d%02P%7b#24%04%01X%1d%02P%a8#25%04%01X%1d%02P%5a#30%04%01X%1d%02P%5b#50%04%01X%1d%02P%7c#90%04%01X%1d%02P%a9#95%04%01X%1d%02P%aa50%04%01X%1d%02P%dc#90%04%01X%1d%02P%e20%04%01X%1d%02$%06

B%1c%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02Q%64400000FF%04%01X%1d%02$%03

B%1d%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02Q%64400000FF%04%01X%1d%02$%05

B%1e%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02Q%64400000FF%04%01X%1d%02$%06

B%1f%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02Q%64400000FF%04%01X%1d%02$%03

B%1f%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02Q%64400000FF%04%01X%1d%02$%03

B%20%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02Q%64400000FF%04%01X%1d%02$%05

B%21%01X%1d%02P%28122)0%04%01X%1d%02P%e20%04%01X%1d%02Q%64400000FF%04%01X%1d%02$%06

B%22%01X%1d%02P%28122)5f%04%01X%1d%02O%6440000000%04%01X%1d%02$%03

B%23%01X%1d%02P%28122)5f%04%01X%1d%02O%6440000000%04%01X%1d%02$%05

B%24%01X%1d%02P%28122)5f%04%01X%1d%02O%6440000000%04%01X%1d%02$%06

14 Appendix: Example CRC16 C Code

```
/* crc16.h
 */

#ifndef     crc16_h
    #define crc16_h

    #include <stdint.h>
    #include <stddef.h>



    #ifdef __cplusplus
        extern "C" {
    #endif



    typedef uint16_t crc_t;



    crc_t crc
        ( crc_t                initialCrc
        , const unsigned char* bufPtr
        , size_t               length
        );



    #ifdef __cplusplus
        } // extern "C"
    #endif



#endif
```

```
/* crc16.c
 */

#include <crc16.h>



crc_t crc
    ( crc_t                initialCrc
    , const unsigned char* p
    , size_t               n
    )
{
      enum
      {
          crcBits  = 16,
          charBits = 8,
          diffBits = crcBits - charBits
      };

      crc_t c = initialCrc;

      #include "crc16tab.h"

      while( n-- )
            c = (c << charBits) ^ crcTab[( c >> diffBits ) ^ *p++];

      return c;
}



/*eof*/
```

```
/* crc16tab.h
 * crc16 table of partial remainders generated by
 * mkcrctab.c with polynomial 1021.
 * included only from within crc() function in file crc16.c
```

```
 */


static const crc_t crcTab[] =
{
    0x0000,  0x1021,  0x2042,  0x3063,  0x4084,  0x50a5,  0x60c6,  0x70e7,
    0x8108,  0x9129,  0xa14a,  0xb16b,  0xc18c,  0xd1ad,  0xe1ce,  0xf1ef,
    0x1231,  0x0210,  0x3273,  0x2252,  0x52b5,  0x4294,  0x72f7,  0x62d6,
    0x9339,  0x8318,  0xb37b,  0xa35a,  0xd3bd,  0xc39c,  0xf3ff,  0xe3de,
    0x2462,  0x3443,  0x0420,  0x1401,  0x64e6,  0x74c7,  0x44a4,  0x5485,
    0xa56a,  0xb54b,  0x8528,  0x9509,  0xe5ee,  0xf5cf,  0xc5ac,  0xd58d,
    0x3653,  0x2672,  0x1611,  0x0630,  0x76d7,  0x66f6,  0x5695,  0x46b4,
    0xb75b,  0xa77a,  0x9719,  0x8738,  0xf7df,  0xe7fe,  0xd79d,  0xc7bc,
    0x48c4,  0x58e5,  0x6886,  0x78a7,  0x0840,  0x1861,  0x2802,  0x3823,
    0xc9cc,  0xd9ed,  0xe98e,  0xf9af,  0x8948,  0x9969,  0xa90a,  0xb92b,
    0x5af5,  0x4ad4,  0x7ab7,  0x6a96,  0x1a71,  0x0a50,  0x3a33,  0x2a12,
    0xdbfd,  0xcbdc,  0xfbbf,  0xeb9e,  0x9b79,  0x8b58,  0xbb3b,  0xab1a,
    0x6ca6,  0x7c87,  0x4ce4,  0x5cc5,  0x2c22,  0x3c03,  0x0c60,  0x1c41,
    0xedae,  0xfd8f,  0xcdec,  0xddcd,  0xad2a,  0xbd0b,  0x8d68,  0x9d49,
    0x7e97,  0x6eb6,  0x5ed5,  0x4ef4,  0x3e13,  0x2e32,  0x1e51,  0x0e70,
    0xff9f,  0xefbe,  0xdfdd,  0xcffc,  0xbf1b,  0xaf3a,  0x9f59,  0x8f78,
    0x9188,  0x81a9,  0xb1ca,  0xa1eb,  0xd10c,  0xc12d,  0xf14e,  0xe16f,
    0x1080,  0x00a1,  0x30c2,  0x20e3,  0x5004,  0x4025,  0x7046,  0x6067,
    0x83b9,  0x9398,  0xa3fb,  0xb3da,  0xc33d,  0xd31c,  0xe37f,  0xf35e,
    0x02b1,  0x1290,  0x22f3,  0x32d2,  0x4235,  0x5214,  0x6277,  0x7256,
    0xb5ea,  0xa5cb,  0x95a8,  0x8589,  0xf56e,  0xe54f,  0xd52c,  0xc50d,
    0x34e2,  0x24c3,  0x14a0,  0x0481,  0x7466,  0x6447,  0x5424,  0x4405,
    0xa7db,  0xb7fa,  0x8799,  0x97b8,  0xe75f,  0xf77e,  0xc71d,  0xd73c,
    0x26d3,  0x36f2,  0x0691,  0x16b0,  0x6657,  0x7676,  0x4615,  0x5634,
    0xd94c,  0xc96d,  0xf90e,  0xe92f,  0x99c8,  0x89e9,  0xb98a,  0xa9ab,
    0x5844,  0x4865,  0x7806,  0x6827,  0x18c0,  0x08e1,  0x3882,  0x28a3,
    0xcb7d,  0xdb5c,  0xeb3f,  0xfb1e,  0x8bf9,  0x9bd8,  0xabbb,  0xbb9a,
    0x4a75,  0x5a54,  0x6a37,  0x7a16,  0x0af1,  0x1ad0,  0x2ab3,  0x3a92,
    0xfd2e,  0xed0f,  0xdd6c,  0xcd4d,  0xbdaa,  0xad8b,  0x9de8,  0x8dc9,
    0x7c26,  0x6c07,  0x5c64,  0x4c45,  0x3ca2,  0x2c83,  0x1ce0,  0x0cc1,
    0xef1f,  0xff3e,  0xcf5d,  0xdf7c,  0xaf9b,  0xbfba,  0x8fd9,  0x9ff8,
    0x6e17,  0x7e36,  0x4e55,  0x5e74,  0x2e93,  0x3eb2,  0x0ed1,  0x1ef0,
};


/*eof*/
```

FACTORY AUTOMATION – SENSING YOUR NEEDS

Pepperl+Fuchs sets the standard in quality and innovative technology for the world of automation. Our expertise, dedication, and heritage of innovation have driven us to develop the largest and most versatile line of industrial sensor technologies and interface components in the world. With our global presence, reliable service, and flexible production facilities, Pepperl+Fuchs delivers complete solutions for your automation requirements – wherever you need us.

Contact
Pepperl+Fuchs GmbH
Lilienthalstraße 200
68307 Mannheim · Germany
Tel. +49 621 776-4411 · Fax +49 621 776-27-4411
E-mail: fa-info@pepperl-fuchs.com

Worldwide Headquarters
Pepperl+Fuchs GmbH · Mannheim · Germany
E-mail: fa-info@pepperl-fuchs.com

USA Headquarters
Pepperl+Fuchs Inc. · Twinsburg, OH · USA
E-mail: fa-info@us.pepperl-fuchs.com

Asia Pacific Headquarters
Pepperl+Fuchs Pte Ltd · Singapore
Company Registration No. 199003130E
E-mail: fa-info@sg.pepperl-fuchs.com

www.pepperl-fuchs.com

PEPPERL+FUCHS
SENSING YOUR NEEDS

/ TDOCT1981__ENG
07/2010